Doing Computational Phonology

September 17, 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 (2008, 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

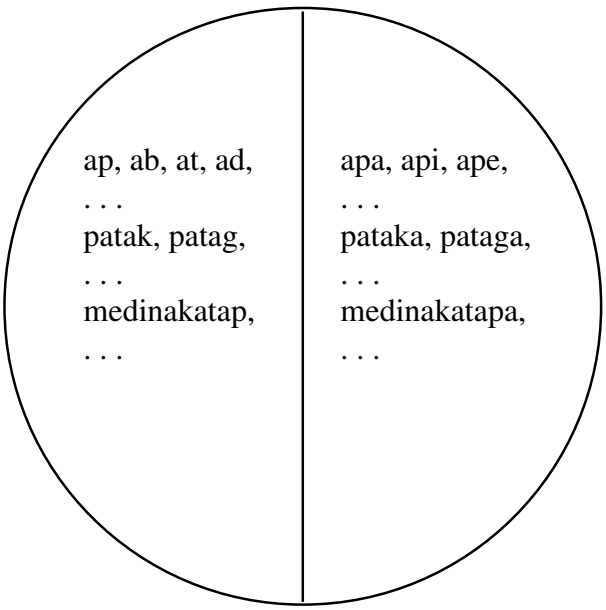


Figure 1.1: The generalization that "Word final vowels are prohibited" partitions the set of all possible forms into two sets.

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

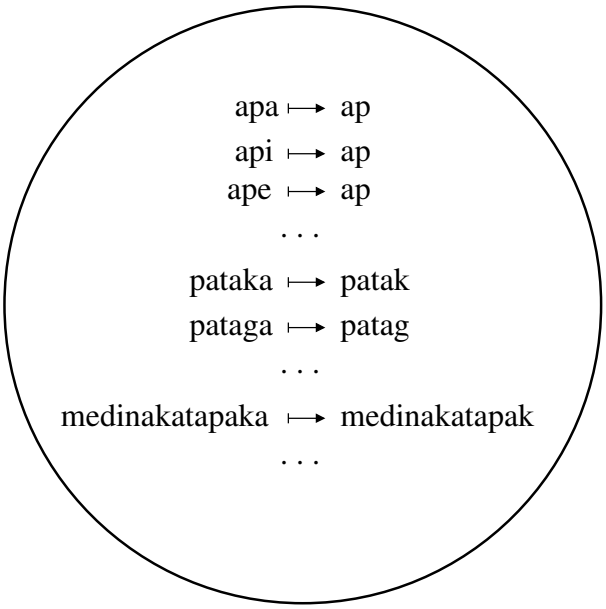


Figure 1.2: The function corresponding to the generalizations that "Word final vowels delete."

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 Kartunnen, 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 (2008, 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*

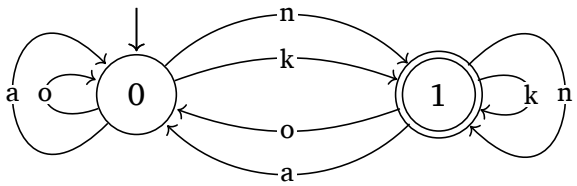


Figure 1.3: A finite state acceptor for the generalization "Word final vowels are prohibited." A simple alphabet {n,k,a,o} is assumed.

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	$\rightarrow$	1	$\rightarrow$	0	$\rightarrow$	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

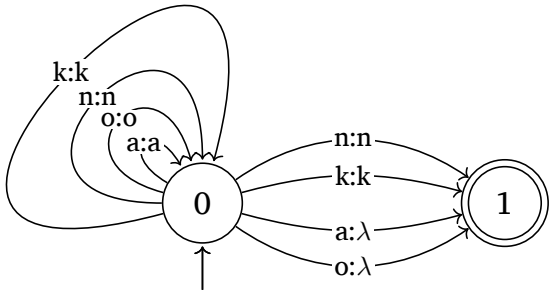


Figure 1.4: A finite state trasnsducer for the generalization "Word final vowels delete." A simple alphabet {n,k,a,o} is assumed.

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

Input:		n		a		k		o	
States:	0	$\rightarrow$	0	$\rightarrow$	0	$\rightarrow$	0	$\rightarrow$	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.

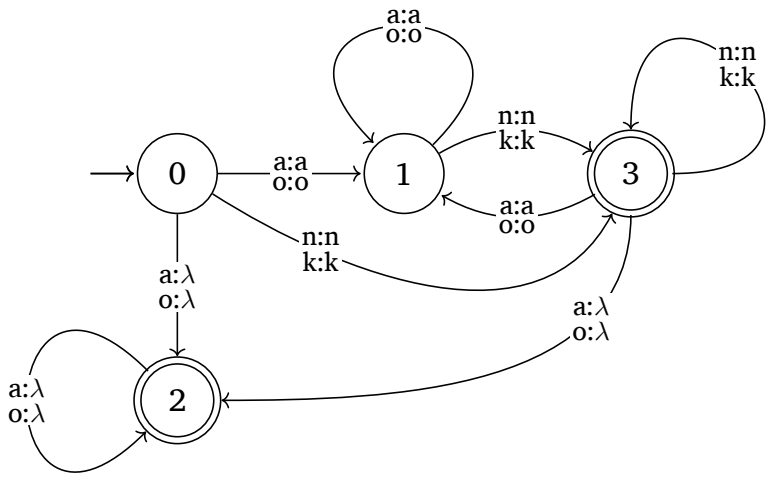


Figure 1.5: A finite state transducer for the generalization "Strings of vowels word-finally delete." A simple alphabet {n,k,a,o} is assumed.

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].

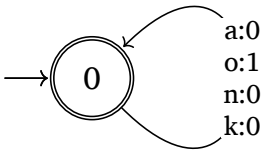


Figure 1.6: A finite state transducer which counts the number of *o*s in words. A simple alphabet {n,k,a,o} is assumed.

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 Kartunnen, 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.

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

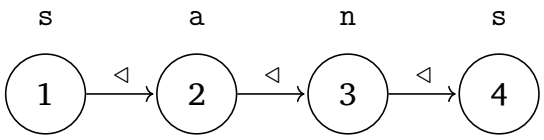


Figure 2.1: At left, the successor model of the word *sans*. At right, a graphical diagram of this model.

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 η.
(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.

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

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

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

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

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

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

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

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

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

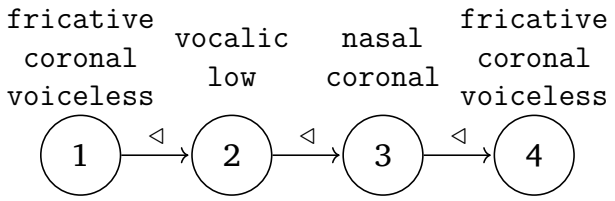


Figure 2.2: At left, the successor model with features of the word *sans*. Unary relations which equal the empty set are omitted for readability. At right, a graphical diagram of this model.

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 $\texttt{closed}\,(X)$.

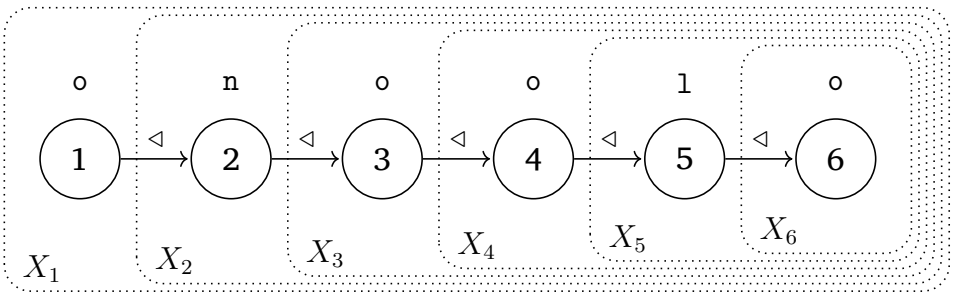


Figure 2.3: The successor model for the word $onool o$. The dotted rectangular regions indicate the sets of domain elements (X_i) which are closed under successor.

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($\lhd$).

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-

$w = \rtimes a^k (n a^k)^{2t} a^k \ltimes$			
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	
$v = \rtimes a^k (n a^k)^{2t} n a^k \ltimes$			
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 $w = a^k (n a^k)^{2t} a^k$ and $v = a^k (n a^k)^{2t} n a^k$ with word boundaries.

cate `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.

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

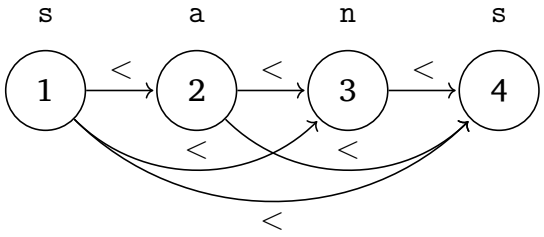


Figure 2.4: At left, the precedence model of the word *sans*. At right, a graphical diagram of this model.

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 (2012a).

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, 2020).

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

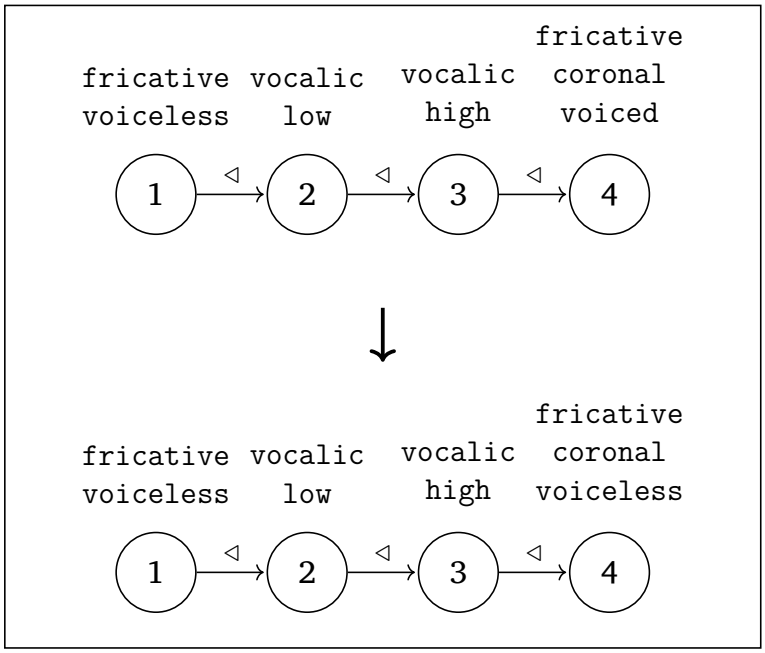


Figure 3.1: A graphical diagram of the feature-based successor model of *hauz* being mapped to the feature-based successor model of *haus*.

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.

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

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.

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

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, 2008). 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

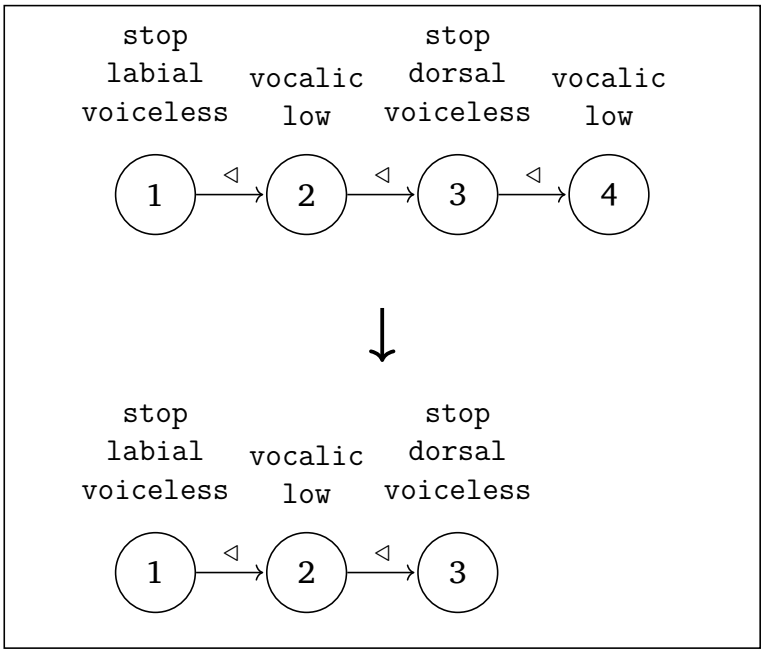


Figure 3.2: A graphical diagram of the feature-based successor model of *paka* being mapped to the feature-based successor model of *pak*.

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.

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

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

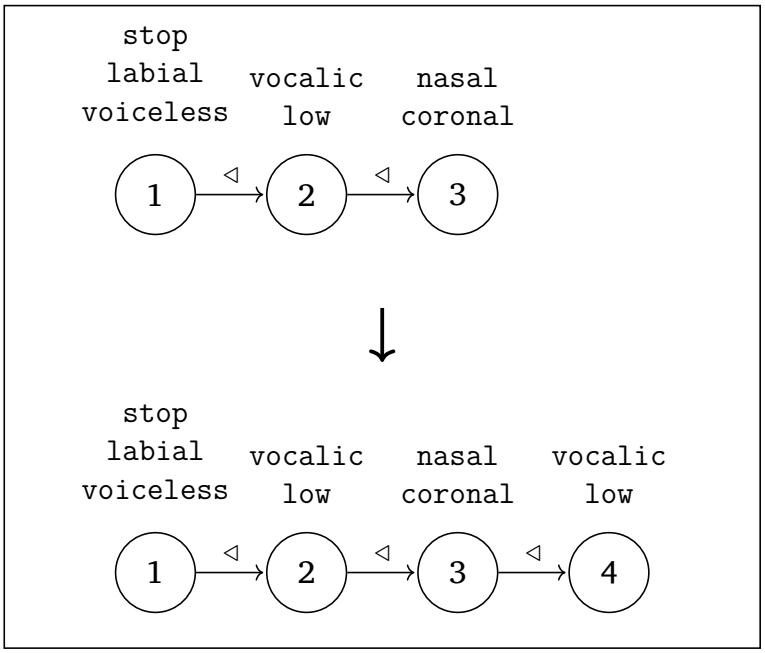


Figure 3.3: A graphical diagram of the feature-based successor model of *pan* being mapped to the feature-based successor model of *pana*.

$\phi^1_{\texttt{license}}(x)$ 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

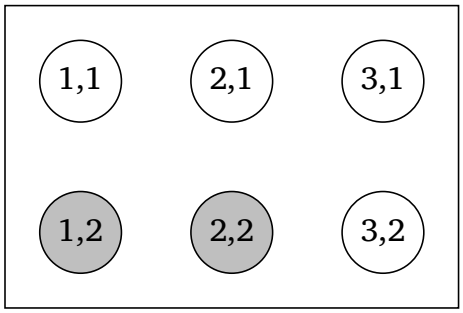


Figure 3.4: The possible domain elements of the output structure for input *pan* when the copy set $C \stackrel{\text{def}}{=} \{1, 2\}$. The unlicensed elements are colored gray.

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

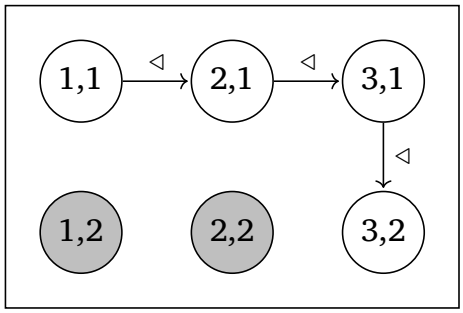


Figure 3.5: The successor relations in the output structure for input *pan* when the copy set $C \stackrel{\text{def}}{=} \{1, 2\}$. The unlicensed elements are colored gray.

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 $\phi^2_{\texttt{R}}(x) \stackrel{\text{def}}{=} \texttt{false}$.

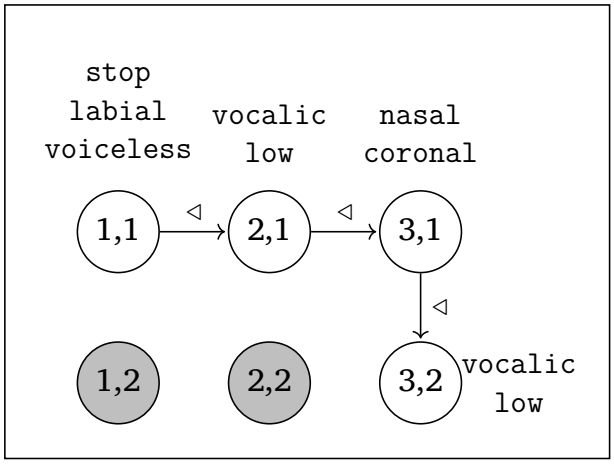


Figure 3.6: The model representing *pana* which is output for the input *pan*. The unlicensed elements are colored gray.

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.

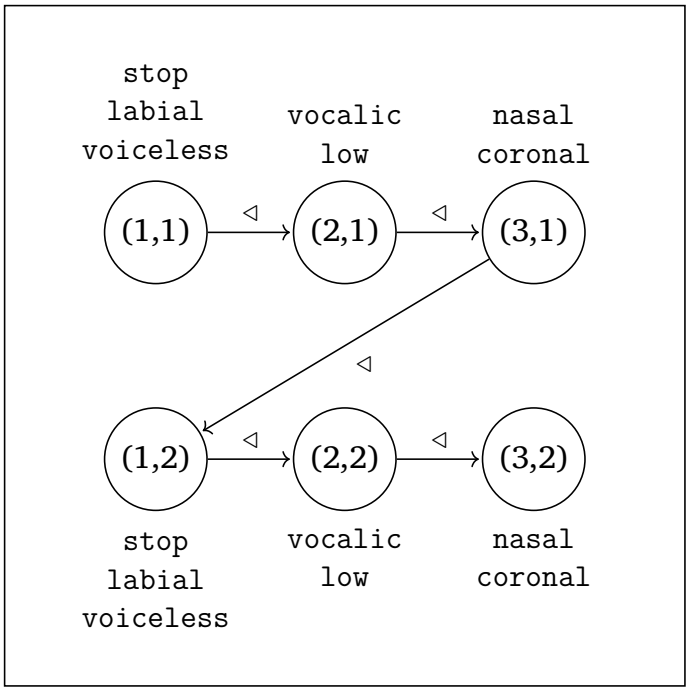


Figure 3.7: The model for *panpan*, which is the output of the reduplication process applying to the input *pan*.

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*.

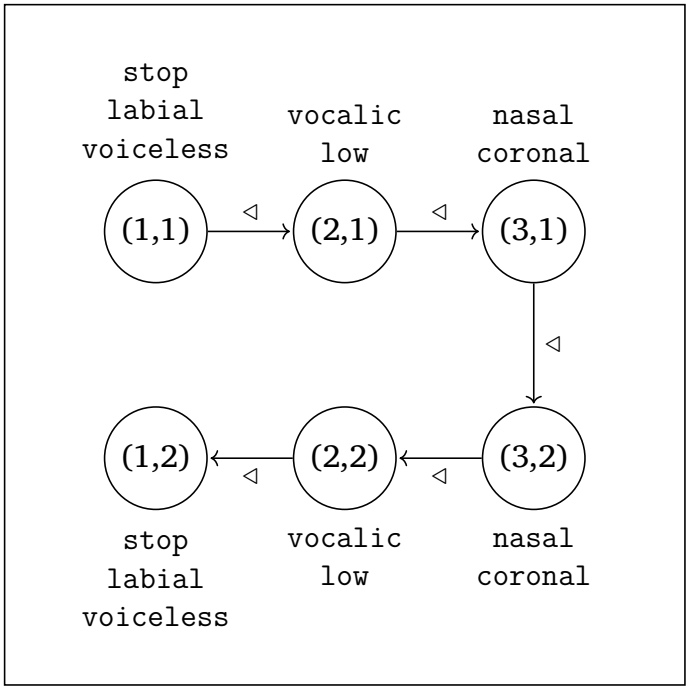


Figure 3.8: The model representing *pannap*, which is the output of the mirroring process applied to the input *pan*.

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,

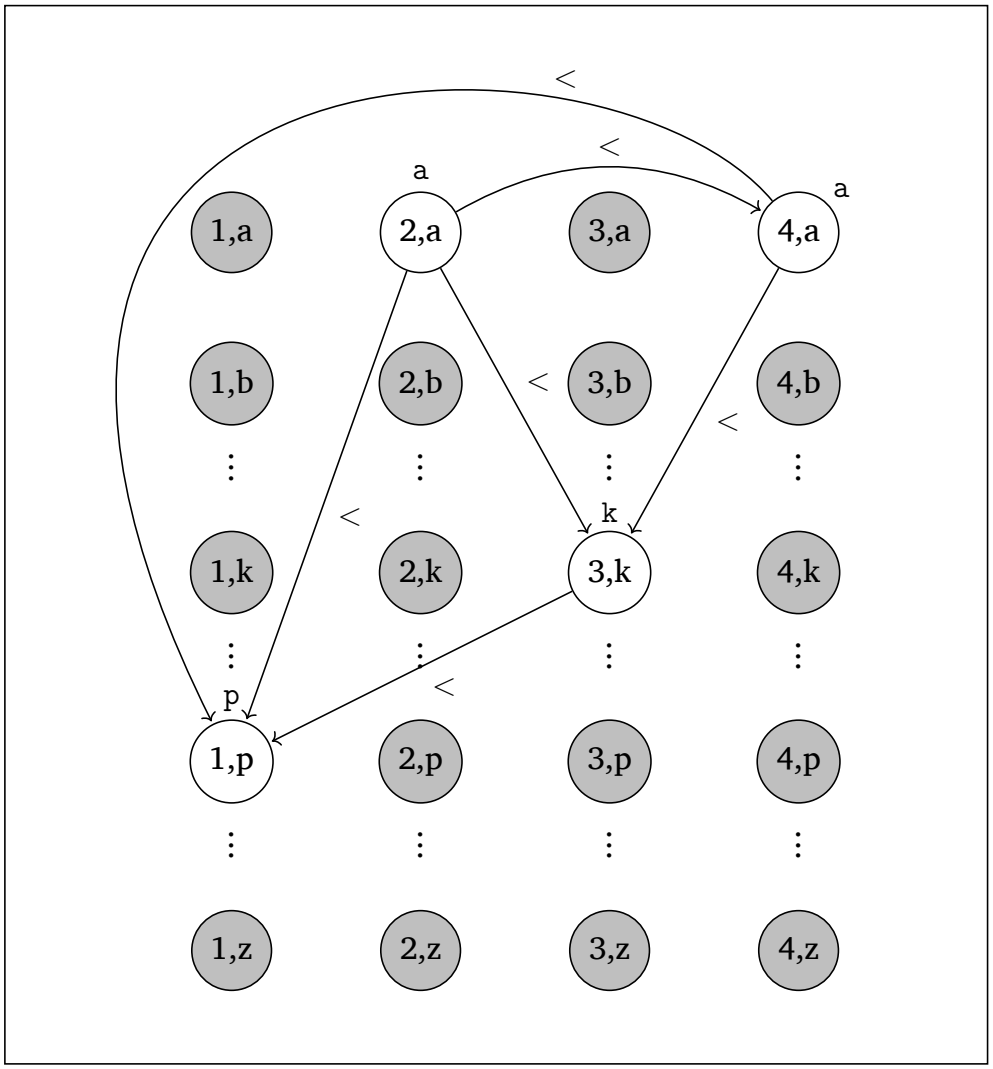


Figure 3.9: The model representing the output *aakp* of the sorting process applied to input *paka*. All potential domain elements shown. Unlicensed elements are in gray. Unary relations are only shown on licensed elements.

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 **??** 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,

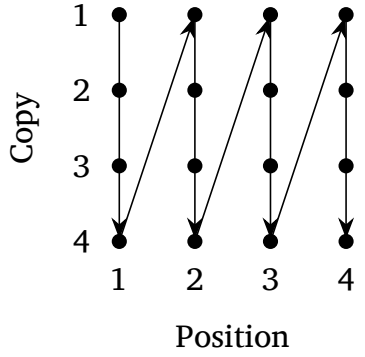


Figure 3.10: Order preservation

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.

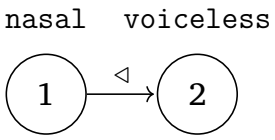


Figure 5.1: The factor NT

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

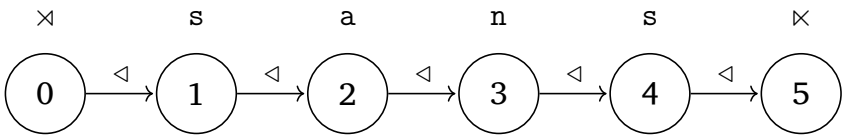


Figure 5.2: A graphical depiction of the successor model with word boundaries of the word *sans*.

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.

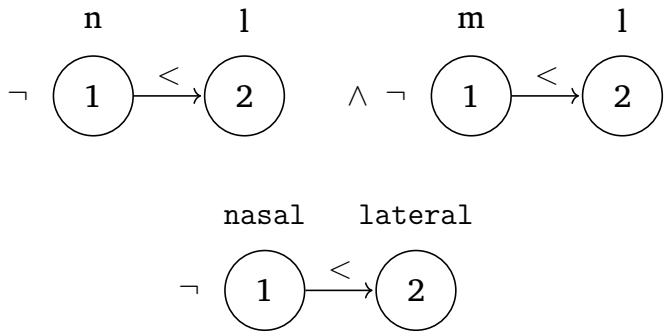


Figure 5.3: The factor NL with letters (above) and features (below)

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 **??**, 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 (1972, 2001); Courcelle (1994); Engelfriet and Hoogeboom (2001); Hedman (2004) and Courcelle and Engelfriet (2012b). 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 rózɨ	u rózɨ	'rose'
	at mʲɛnɨ	bʲiz mʲɛnɨ	u mʲɛnɨ	'change'
	at lunɨ	bʲiz lunɨ	u lunɨ	'moon'
	at álɨ	bʲiz álɨ	u álɨ	'Ala' (name)
	at irɨ	bʲiz irɨ	u irɨ	'Ira' (name)
voiced obstruent	ad valnɨ	bʲiz valnɨ	u valnɨ	'wave'
	ad baradɨ	bʲiz baradʹ	u baradʹ	'beard'
	ad galavɨ	bʲiz galavɨ	u galavɨ	'head'
voiceless obstruent	at pʲitɨ	bʲis pʲitɨ	u pʲitɨ	'heel'
	at sʲistrʹ	bʲis sʲistrʹ	u sʲistrʹ	'sister'
	at karovɨ	bʲis karóvɨ	u karóvɨ	'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 valnɨ/	/at pʲitɨ/	/bʲiz valnɨ/	/bʲiz pʲitɨ/
Assimilation	ad valnɨ			bʲis pʲitɨ
Output SR	[ad valnɨ]	[at pʲitɨ]	[bʲiz valnɨ]	[bʲis pʲitɨ]

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

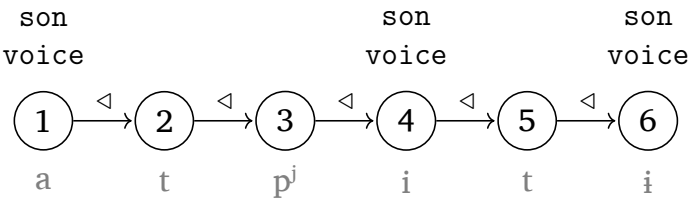
This section provides an analysis using First Order logic and model-theoretic representations of regressive voicing assimilation in Russian obstruent clusters. It is important to be clear both about the representations and the transformations.

7.3.1 Representations

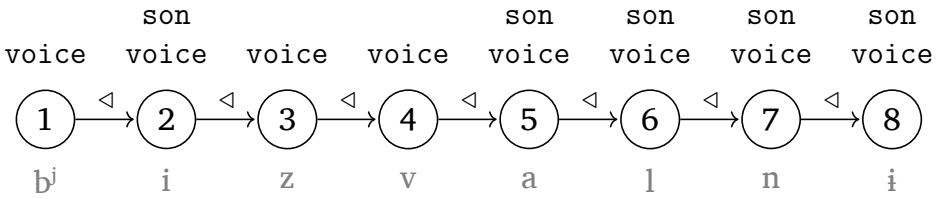
Voicing assimilation has two key properties: a) it targets the features [$\pm$voice] and [$-$son], b) it references adjacent segments. Therefore the input and output word models must be able to reference these properties.

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.

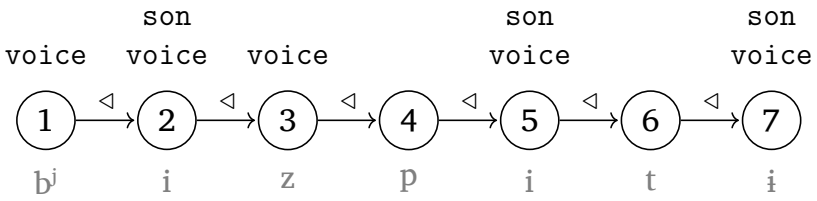
$$\mathfrak{R} = \{\triangleleft\} \cup \mathcal{F}$$



(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 ◁.

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 $\mathtt{f} \in \mathcal{F}$ 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.

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

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

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

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

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

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

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

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,

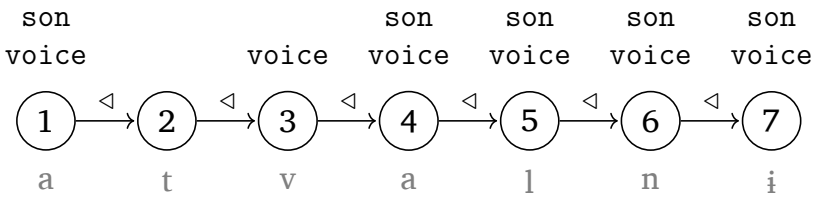
(b) it is a voiced obstruent that does not precede a voiceless obstruent, or

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

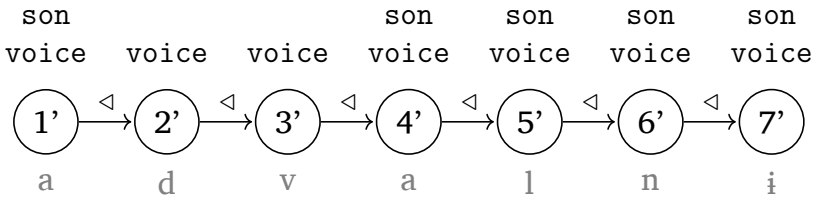
These three conditions are captured via disjunction in the output formula below.

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

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, 2020) 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.

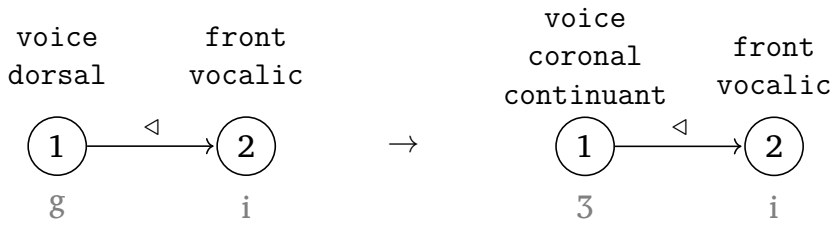


Figure 8.1: Word models for the input /gi/ and output [ʒi].

	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.

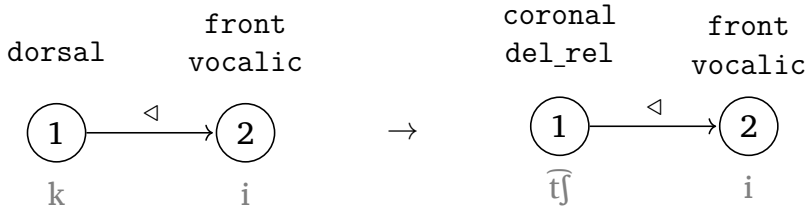


Figure 8.2: Word models for the input /ki/ and output [t͡ʃi].

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].

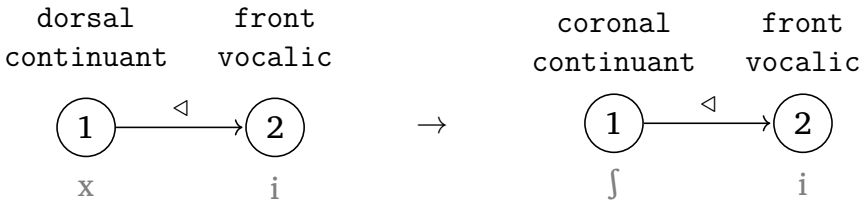


Figure 8.3: Word models for the input /xi/ and output [ʃi].

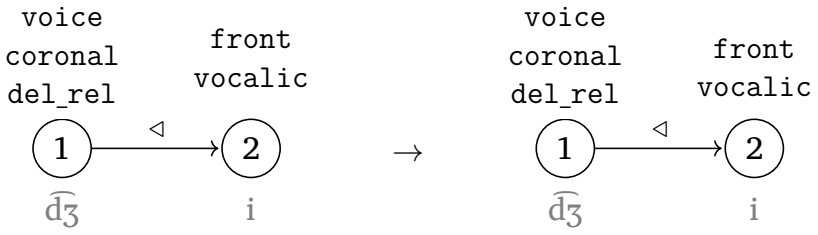


Figure 8.4: Word model for the input /d͡ʒi/ and the output [d͡ʒi].

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.

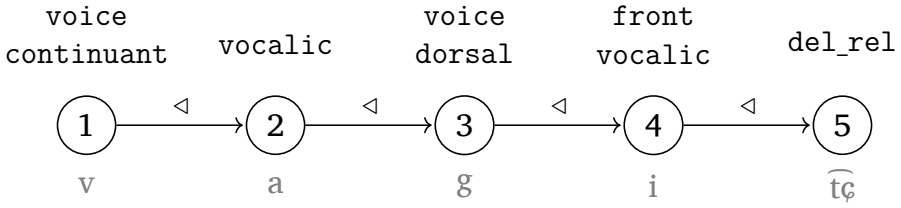


Figure 8.5: Word model for the input /vagi͡tɕ/

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].

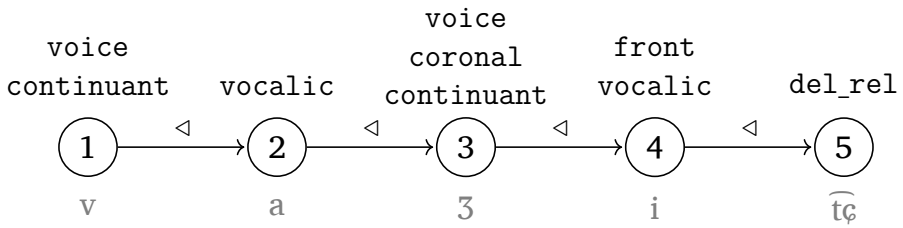


Figure 8.6: Word model for the output [vaʒit͡ɕ]

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.

$$\mathfrak{R} = \{\texttt{voc}, \texttt{low}, \texttt{high}, \texttt{back}, \texttt{delrel}, \texttt{stop}, \texttt{nas}, \texttt{lat}, \texttt{appr}, \texttt{lab}, \texttt{cor}, \texttt{dist}, \texttt{dor}, \triangleleft\} \tag{9.1}$$

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.

$$\phi_{\texttt{domain}} \stackrel{\text{def}}{=} \texttt{true} \tag{9.2}$$

$$C \stackrel{\text{def}}{=} \{1\} \tag{9.3}$$

$$\phi_{\texttt{license}} \stackrel{\text{def}}{=} \texttt{true} \tag{9.4}$$

$$\phi_{\triangleleft}(x, y) \stackrel{\text{def}}{=} x \triangleleft y \tag{9.5}$$

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)’.

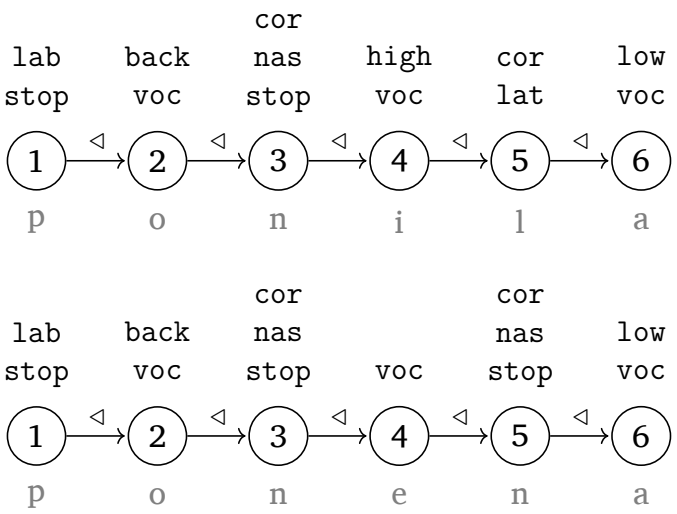


Figure 9.2: Illustrating the input and output structures corresponding to underlying /ponila/ surfacing as [ponena] ‘fall (applied)’.

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.

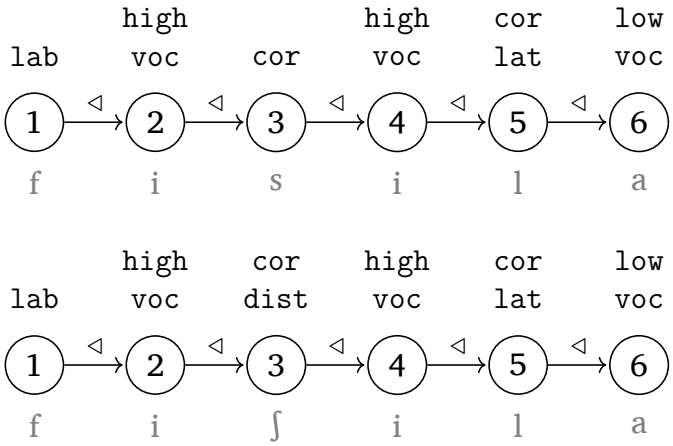


Figure 9.3: Illustrating the input and output structures corresponding to underlying /fisila/ surfacing as [fiʃila] 'hide (applied).'

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).

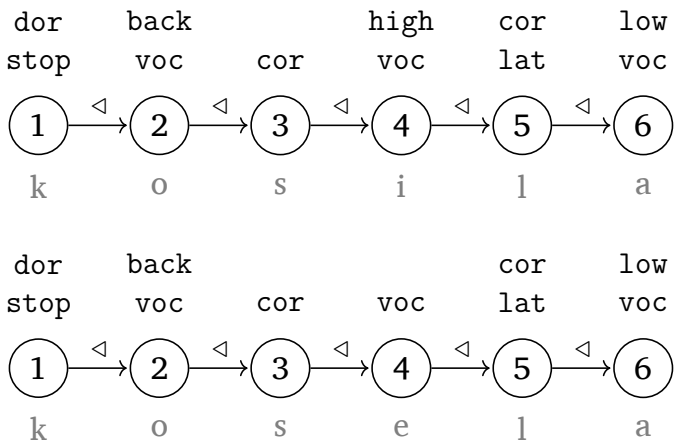


Figure 9.4: Illustrating the input and output structures corresponding to underlying /kosila/ surfacing as [kosela] 'hide (applied).'

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, 2012b). 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.

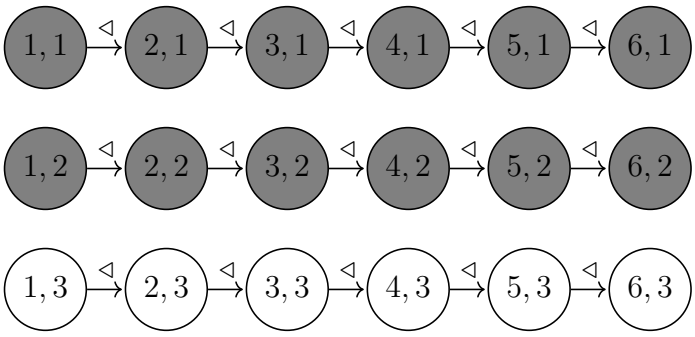


Figure 9.5: The effect of the copyset and the licensing and successor formulas on the output structure of an input word of length six. Unlicensed domain elements are grayed out. Element (x,y) refers to the yth copy of x. Unary relations are omitted.

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.

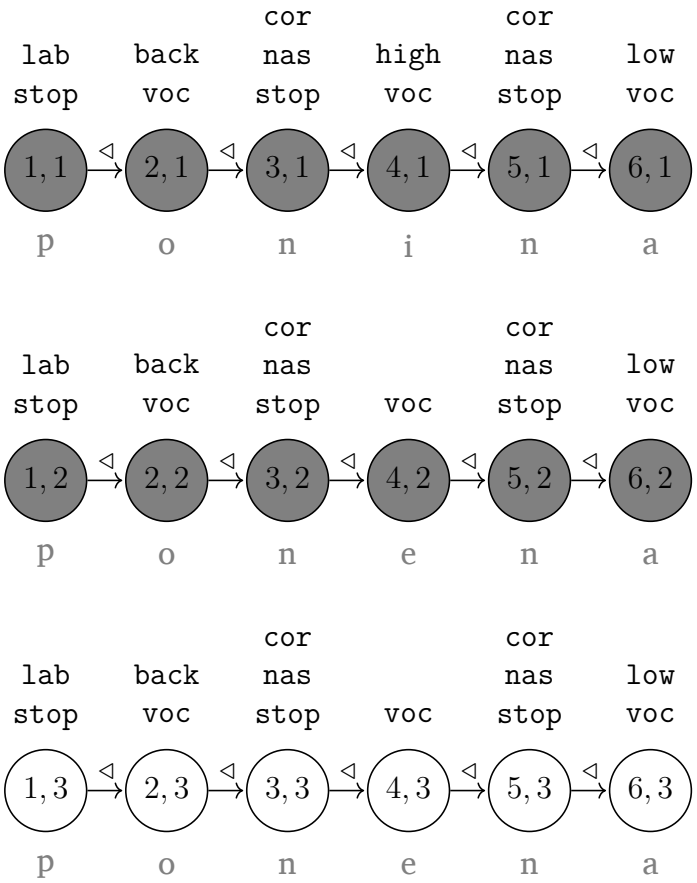


Figure 9.6: The first, second and third copies illustrating the logical transduction which maps /ponila/ to [ponena] 'fall (applied).'

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.

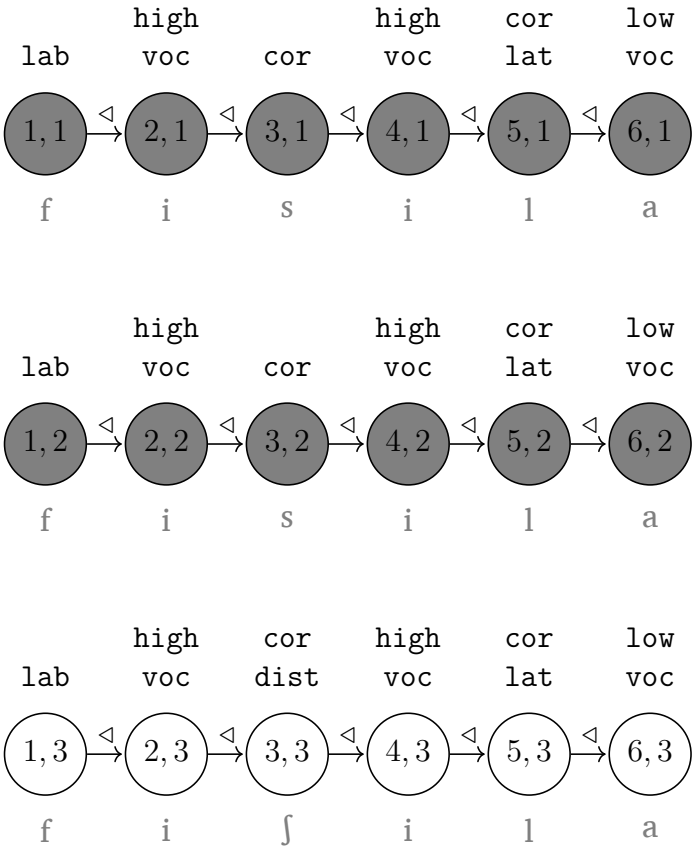


Figure 9.7: The first, second and third copies illustrating the logical transduction which maps /fisila/ to [fiʃila] 'hide.'

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).

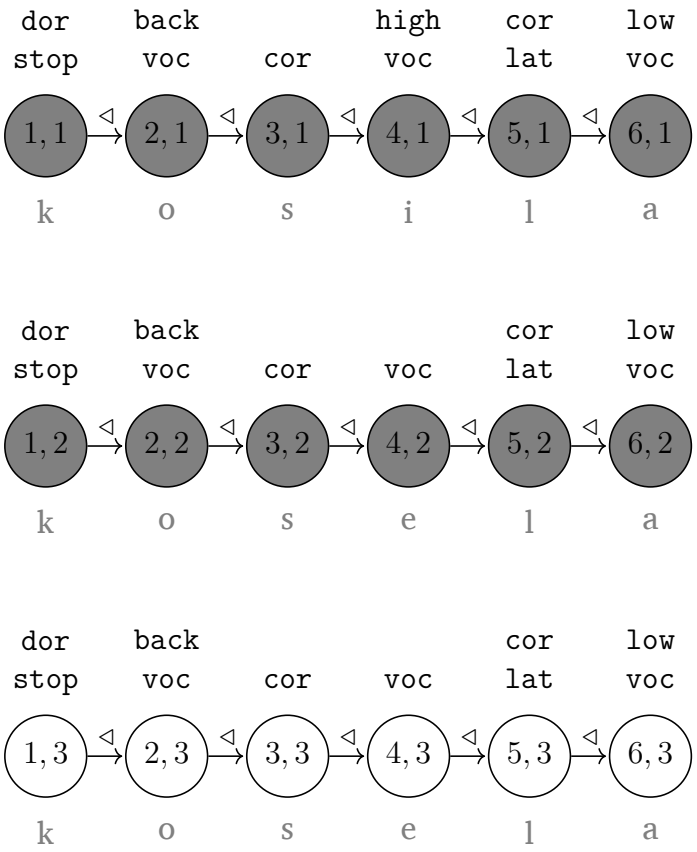


Figure 9.8: The first, second and third copies illustrating the logical transduction which maps /kosila/ to [kosela] 'be strong (applied).'

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
$\texttt{voc(x)}$	$\bot$	$\top$	$\bot$	$\top$
$\texttt{g}\,(x)$	$\top$	$\bot$	$\top$	$\bot$
$\texttt{VgV}\,(x)$	$\bot$	$\bot$	$\top$	$\bot$
$\phi_{\texttt{license}}(x)$	$\top$	$\top$	$\bot$	$\top$

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.

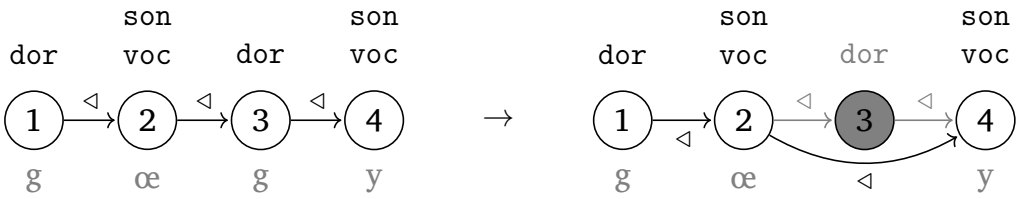


Figure 10.1: Visualization of /gœg-y/→[gœ-y]. The only unary relations shown are `dor`, `son`, and `voc`. Unlicensed domain elements, and their associated relations, are grayed out.

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.'

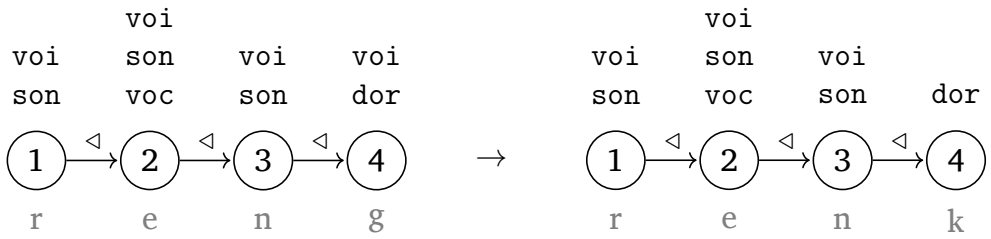


Figure 10.2: /reng/→[renk]

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.'

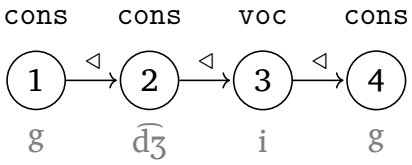


Figure 11.1: A visualization of the $\mathfrak{U}$-structure representing /gd͡ʒig/ 'one.' Only features `cons` and `voc` are shown.

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.

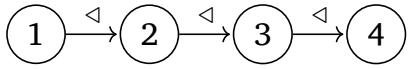


Figure 11.2: The binary relations that would be present in the output structure if every domain element is licensed.

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.

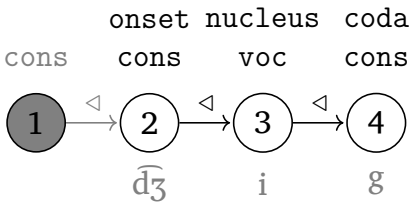


Figure 11.3: The relational structure produced by the input /d͡ʒig/ 'one' shown in Figure 11.1 under the transductions defined in Equations 11.3 through 11.10. Unlicensed elements and their associated relations are grayed out.

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.

Underlying Form	ku - ɣoʃo - a	H ku - fumbatiʃ - a
Surface Form	ku - ɣoʃo - a 'to do'	H ku - fumbatiʃ - a 'to tie securely'
Underlying Form	H ku - vi - ɣoʃo - a	H ku - fumbatiʃ- ij - an - a
Surface Form	H ku - vi - ɣoʃo - a 'to do them'	H ku - fumbatiʃ- ij - an - a '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.)

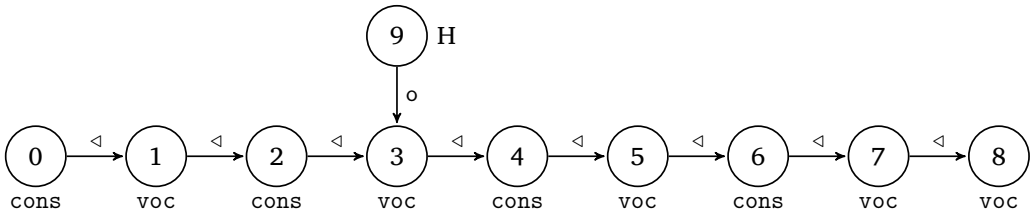


Figure 12.1: A graph representing the autosegmental model of Shambaa /ku-ví-ɣoʃo-a/ 'to do them'. Note both the edges representing successor $\triangleleft$ and association $\circ$.

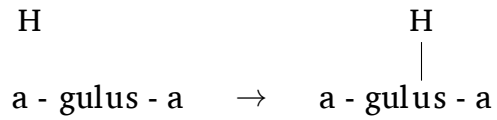
The graph in Figure 12.1 has the usual directed edges representing the successor relation $\triangleleft$ among segments. Additionally, there is an association $\circ$ between the domain element 9, which bears the H tone, and domain element 3, which corresponds to the second vowel.

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.

$$\varphi_{\triangleleft}(x) \stackrel{\text{def}}{=} x \triangleleft y \tag{12.17}$$

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 $\triangleleft_V$ of vowels, ignoring consonants.

$$\begin{aligned} x \mathrel{\triangleleft_{\texttt{V}}} y \stackrel{\text{def}}{=} \quad & \texttt{voc}(x) \wedge \texttt{voc}(y) \wedge x \leq y \\ & \wedge (\forall z)[(x \leq z \leq y) \rightarrow \neg \texttt{voc}(z)] \end{aligned} \tag{12.18}$$

In other words, $x \mathrel{\triangleleft_{\texttt{V}}} y$ 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 $\triangleleft_{\texttt{V}}$.

$$\texttt{finalV}\,(x) \stackrel{\text{def}}{=} \quad \texttt{voc}(x) \wedge \neg(\exists y)[x \mathrel{\triangleleft_{\texttt{V}}} y] \tag{12.19}$$

The penultimate vowel is then the vowel that precedes the final vowel with respect to $\triangleleft_{\texttt{V}}$.

$$\phi_{\texttt{penultV}}(x) \stackrel{\text{def}}{=} \quad (\exists y)[x \mathrel{\triangleleft_{\texttt{V}}} y \wedge \texttt{finalV}\,(y)] \tag{12.20}$$

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’.

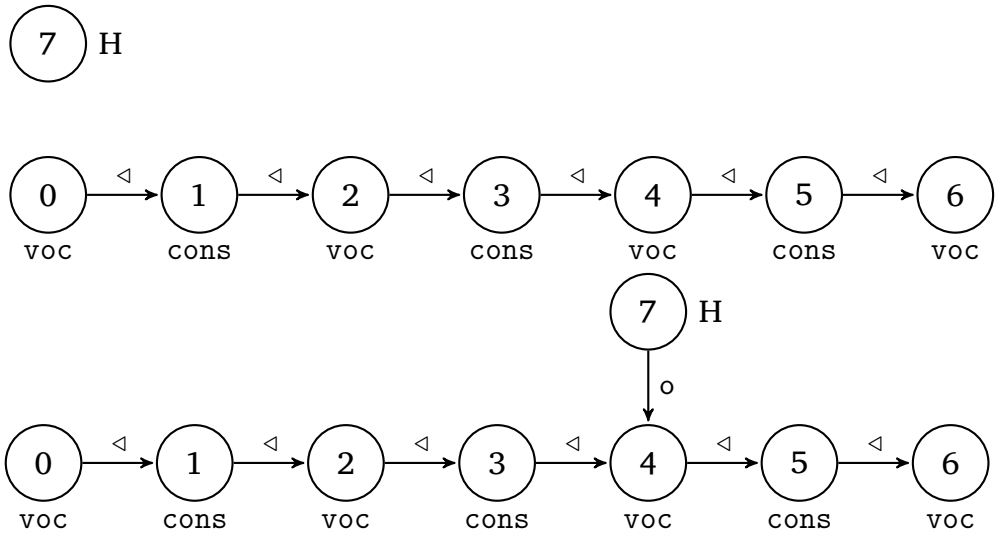


Figure 12.3: Visualization of the transduction of the autosegmental representations of Zigula /a$^{\text{H}}$-gulus-a/ $\rightarrow$ [a-gulús-a] 'he/she is chasing'.

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'.

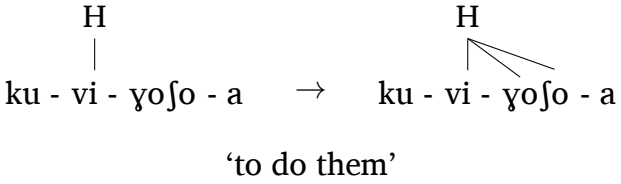


Figure 12.4: An example of unbounded spreading in Shambaa

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'.

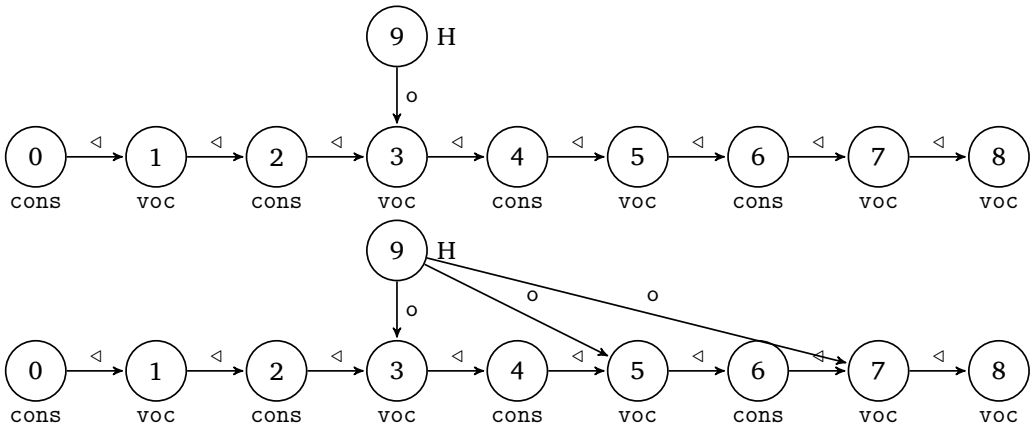


Figure 12.5: Visualization of the transduction of the autosegmental representation of Shambaa 'he/she is chasing'/ku-ví-ɣoʃo-a/→[ku-ví-ɣóʃó-a] 'to do them'.

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.

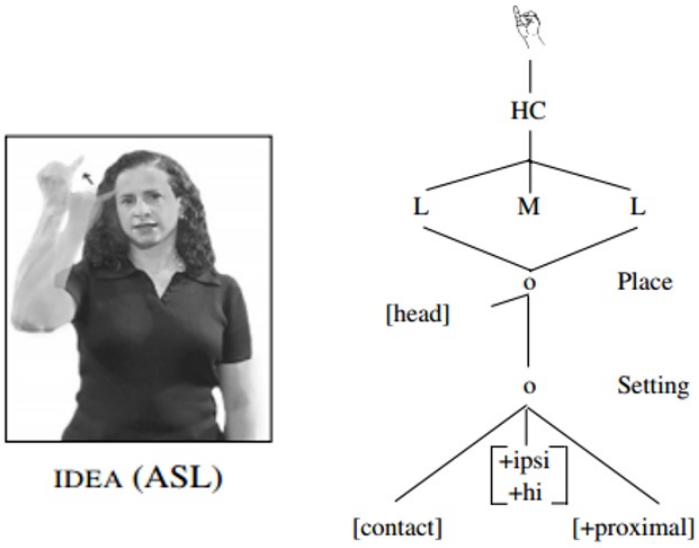


Figure 13.1: The ASL sign 'IDEA' (left) and its Hand Tier Model (right) (Images copyright Wendy Sandler & Diane Lillo-Martin).

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.

$$\texttt{sign\_ASR} \stackrel{\text{def}}{=} \texttt{separability} \wedge \texttt{good\_tiers} \wedge \texttt{good\_associations} \wedge \texttt{NCC} \tag{13.2}$$

$$
\begin{aligned}
\langle \mathcal{D} &= \{1,2,3,4,5\} \\
\mathtt{l}_1 &= \{1,3\} \\
\mathtt{m}_1 &= \{2\} \\
\mathtt{h}_1 &= \{4\} \\
\mathtt{p}_1 &= \{5\} \\
&\ldots \\
<&= \{\langle 1,2\rangle, \langle 1,3\rangle, \langle 2,3\rangle\} \\
\mathtt{A} &= \{\langle 1,4\rangle, \langle 2,4\rangle, \langle 3,4\rangle\} \\
\mathtt{loc} &= \{\langle 1,5\rangle, \langle 3,5\rangle\}\rangle
\end{aligned}
$$

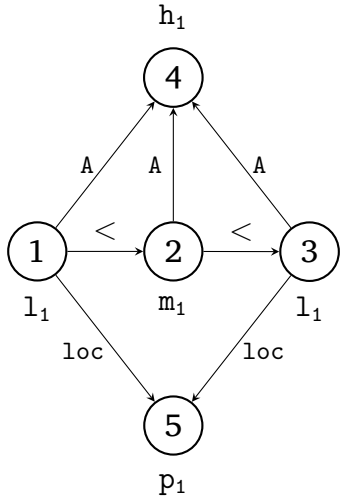


Figure 13.2: Autosegmental Hand Tier word model of a monosyllabic sign and a visualization. For all $n \neq 1$, unary relations $\mathtt{l}_n \in L$, $\mathtt{m}_n \in M$, $\mathtt{h}_n \in H$, and $\mathtt{p}_n \in P$ equal the empty set and are omitted for readability. Additionally, only some elements 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.

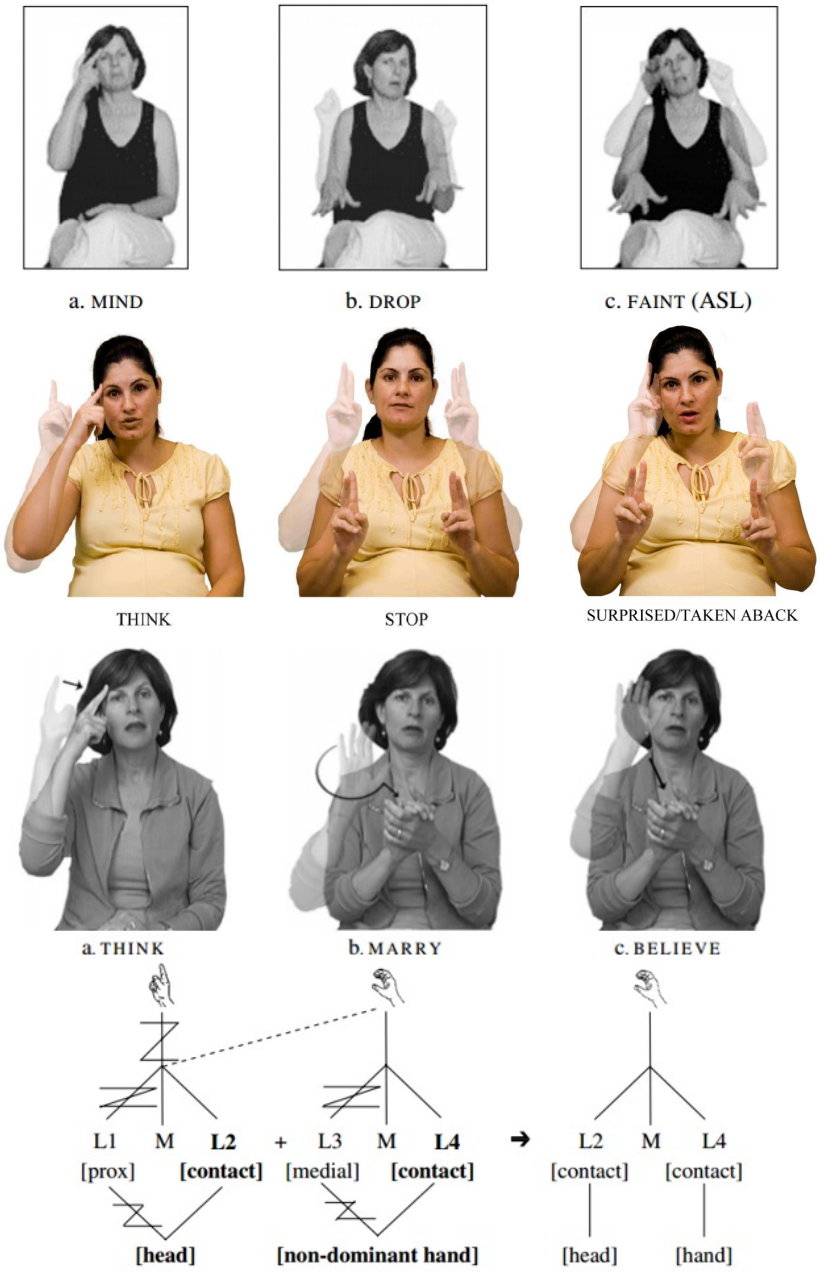


Figure 13.3: Compound Reduction. Top: ASL MIND + DROP = FAINT; Middle: ISL THINK + STOP = SURPRISED; Bottom: ASL THINK + MARRY = BELIEVE, with Hand Tier model of reduction (Images copyright Wendy Sandler & Diane Lillo-Martin).

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.

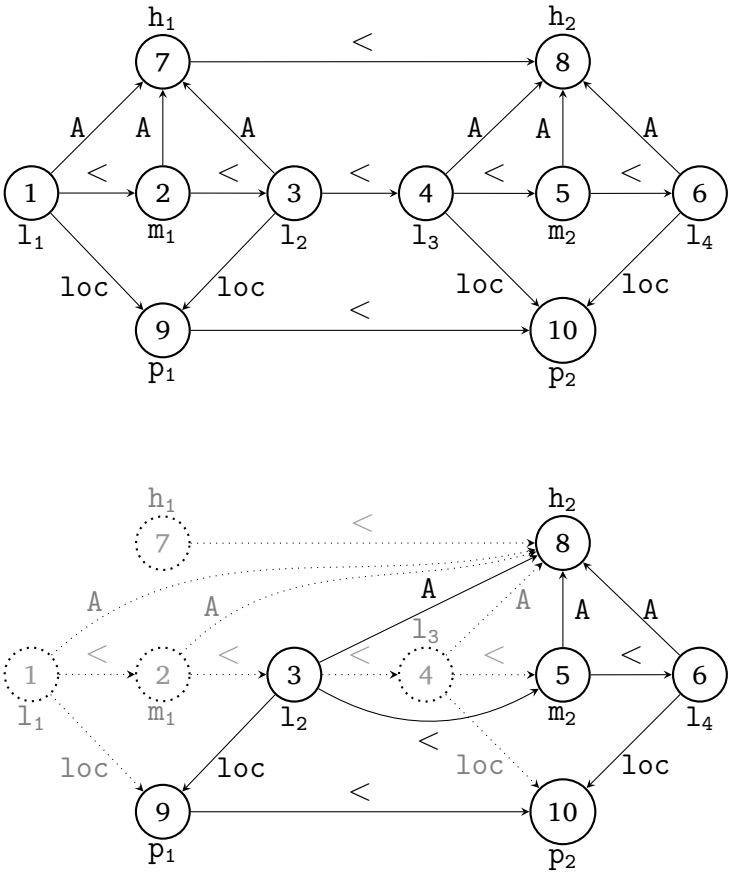


Figure 13.4: Visual of Input (top) and Output (bottom) $\mathfrak{R}$-structures for compound reduction of ASL BELIEVE (THINK + MARRY). Unlicensed elements and their associated relational elements are dashed and in gray. Only some elements of the precedence relation are shown for readability.

	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 **??** provide a logical characterization of this class and Strother-Garcia (2018) and Chandlee and Jardine (2019) 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.'

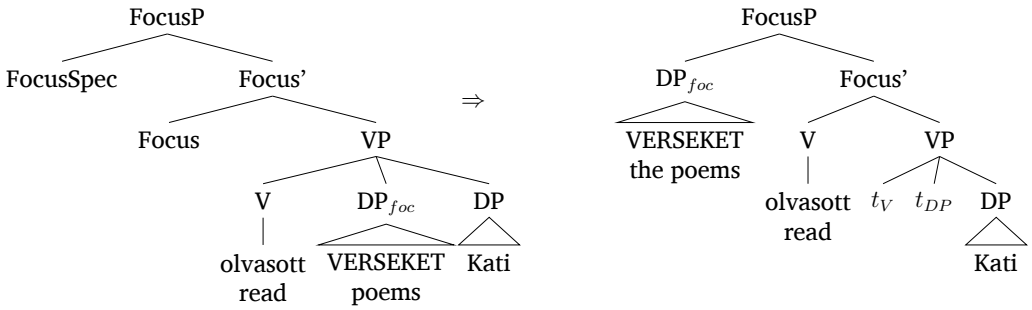


Figure 14.1: Base tree and derived tree illustrating focus and verb movement in Hungarian.

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.

$$\texttt{FP}\ (x) \stackrel{\text{def}}{=} \texttt{Focus}(x) \wedge \texttt{phrase}(x) \tag{14.2}$$

$$\texttt{FS}\ (x) \stackrel{\text{def}}{=} \texttt{Focus}(x) \wedge \texttt{specifier}(x) \tag{14.3}$$

$$\overline{\texttt{F}}\ (x) \stackrel{\text{def}}{=} \texttt{Focus}(x) \wedge \texttt{bar}(x) \tag{14.4}$$

$$\texttt{FH}\ (x) \stackrel{\text{def}}{=} \texttt{Focus}(x) \wedge \texttt{head}(x) \tag{14.5}$$

$$\texttt{VP}\ (x) \stackrel{\text{def}}{=} \texttt{V}(x) \wedge \texttt{phrase}(x) \tag{14.6}$$

$$\texttt{VH}\ (x) \stackrel{\text{def}}{=} \texttt{V}(x) \wedge \texttt{head}(x) \tag{14.7}$$

$$\texttt{DP}\ (x) \stackrel{\text{def}}{=} \texttt{D}(x) \wedge \texttt{phrase}(x) \tag{14.8}$$

$$\texttt{DP}_{\texttt{foc}}\ (x) \stackrel{\text{def}}{=} \texttt{DP}(x) \wedge \texttt{focused}(x) \tag{14.9}$$

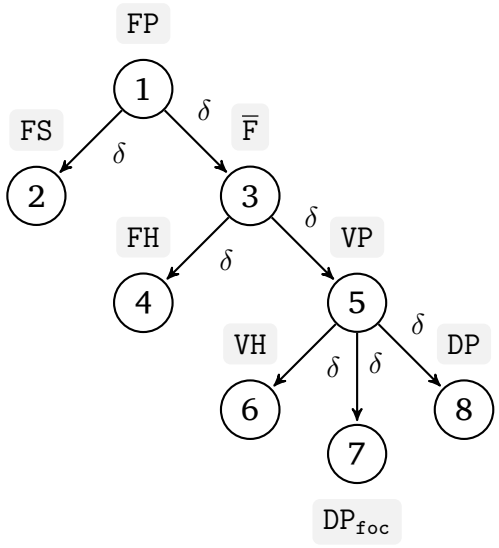


Figure 14.2: Model of input tree in Figure 14.1. The successor relation among siblings is omitted for readability.

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.

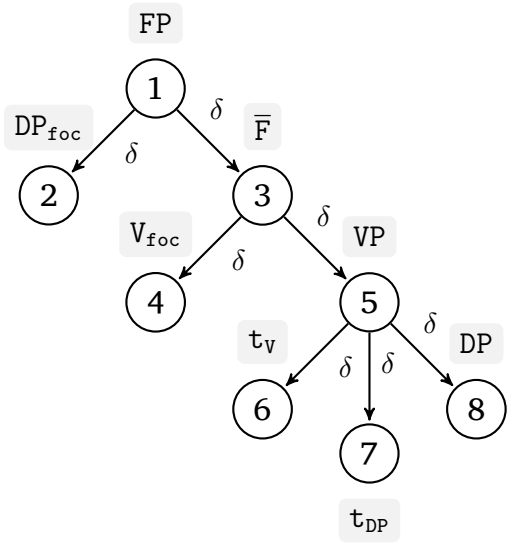


Figure 14.3: Model of the output tree in Figure 14.1. The successor relation among siblings is omitted for readability.

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 **??** and **??** 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

Part IV

Horizons

Bibliography

Albro, Dan. 2005. A large-scale, LPM-OT analysis of Malagasy. Doctoral dissertation, University of California, Los Angeles.

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`

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.

Baković, Eric. 2000. Harmony, dominance and control. Doctoral dissertation, Rutgers University.

Bale, Alan, and Charles Reiss. 2018. *Phonology: A Formal Introduction*. The MIT Press.

Beauquier, D., and J.E. Pin. 1991. Languages and scanners. *Theoretical Computer Science* 84:3–21.

Beesley, Kenneth, and Lauri Kartunnen. 2003. *Finite State Morphology*. CSLI Publications.

Benua, Laura. 1997. Transderivational identity: Phonological relations between words. Doctoral dissertation, University of Massachusetts, Amherst.

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.

Chandlee, Jane. 2014. Strictly local phonological processes. Doctoral dissertation, The University of Delaware.

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

Chandlee, Jane, and Adam Jardine. 2019. Autosegmental input-strictly local functions. *Transactions of the Association for Computational Linguistics* 7:157–168.

Chomsky, Noam. 1995. *The Minimalist Program*. The MIT Press.

Chomsky, Noam, and Morris Halle. 1968. *The Sound Pattern of English*. New York: Harper & Row.

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 126:53–75.

Courcelle, Bruno, and Joost Engelfriet. 2012a. *Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach*. Cambridge: Cambridge University Press.

Courcelle, Bruno, and Joost Engelfriet. 2012b. *Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach*. Cambridge University Press.

Dolatian, Hossep. 2020. Computational locality of cyclic phonology in armenian. Doctoral dissertation, Stony Brook University.

Dolatian, Hossep, and Jeffrey Heinz. 2020. Computing and classifying reduplication with 2-way finite-state transducers. *Journal of Language Modelling* 8:179–250.

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.

Eliasson, S. 1985. Turkish k-deletion: simplicity vs. retrieval. *Folia Linguistica* 19:289–312.

Enderton, Herbert B. 1972. *A Mathematical Introduction to Logic*. Academic Press.

Enderton, Herbert B. 2001. *A Mathematical Introduction to Logic*. 2nd ed. Academic Press.

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

Finley, Sara. 2008. The formal and cognitive restrictions on vowel harmony. Doctoral dissertation, Johns Hopkins University, Baltimore, MD.

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.

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.

Golan, Jonathan S. 1999. *Semirings and their Applications*. Springer.

Goldsmith, John. 1976. Autosegmental phonology. Doctoral dissertation, MIT, Cambridge, MA.

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. 2013. Local and transderivational constraints in syntax and semantics. Doctoral dissertation, University of California, Los Angeles.

Gussmann, Edmund. 2007. *The Phonology Of Polish*. Oxford University Press.

Halle, Morris, and G. N. Clements. 1983. *Problem Book in Phonology*. Cambridge, MA: MIT Press.

Hammond, Michael. 1988. On deriving the well-formedness condition. *Linguistic Inquiry* 19:319–325.

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.

Harris, James. 1983. *Syllable Structure And Stress in Spanish: a Nonlinear Analysis*. Cambridge, Mass.: MIT Press.

Hayes, Bruce. 2009. *Introductory Phonology*. Wiley-Blackwell.

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. 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. Cambridge, UK: Cambridge University Press.

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.

Hopcroft, John, Rajeev Motwani, and Jeffrey Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation*. 3rd ed. Addison-Wesley.

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 12th Conference of the European Chapter of the Association for Computational Linguistics*, 29–32. Association for Computational Linguistics.

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.

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.

Hyman, Larry. 1975. *Phonology: Theory and Analysis*. Holt, Rinehart and Winston.

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.

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.

Kaplan, Ronald, 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 *FSMNLP'98*, 1–12. International Workshop on Finite-State Methods in Natural Language Processing, Bilkent University, Ankara, Turkey.

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.

Kornai, Andras. 1995. *Formal Phonology*. Outstanding Dissertations in Linguistics. Garland Publishing.

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.

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, 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.

Lepic, Ryan. 2015. Motivation in morphology: Lexical patterns in asl and english. Doctoral dissertation, UC San Diego.

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.

Łubowicz, Anna. 2002. Derived environment effects in Optimality Theory. *Lingua* 112:243–280.

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. 2008. *Doing Optimality Theory*. Malden, MA: Blackwell.

McCarthy, John, and Alan Prince. 1995. Faithfulness and reduplicative identity. In *Papers in Optimality Theory*, edited by Jill Beckman, Laura Walsh Dickey, and Suzanne Urbanczyk, no. 18 in University of Massuchusetts Occasional Papers in Linguistics, 249–384.

McNaughton, Robert, and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

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)*.

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`

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.

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.

Perlmutter, David M. 1993. Sonority and syllable structure in american sign language. In *Current issues in ASL phonology*, 227–261. Elsevier.

Postal, Paul M. 1968. *Aspects of Phonological Theory*. Harper & Row.

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.

Rawski, Jonathan. 2017. Phonological complexity is subregular: Evidence from sign language. In *Proceedings of the 53rd Chicago Linguistics Society Annual Meeting*.

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.

Riggle, Jason. 2004. Generation, recognition, and learning in finite state Optimality Theory. Doctoral dissertation, University of California, Los Angeles.

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.
URL `https://www.aclweb.org/anthology/W19-5706`

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.

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.

Savitch, Walter J. 1993. Why it may pay to assume that languages are infinite. *Annals of Mathematics and Artificial Intelligence* 8:17–25.

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.

Shukla, Shaligram. 2000. *Hindi Phonology*. Muenchen: Lincom Europa.

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.

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.

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.

Stowell, Timothy Angus. 1981. Origins of phrase structure. Doctoral dissertation, Massachusetts Institute of Technology.

Strother-Garcia, Kristina. 2018. Imdlawn Tashlhiyt Berber syllabification is quantifier-free. In *Proceedings of the Society for Computation in Linguistics*, vol. 1. Article 16.

Strother-Garcia, Kristina. 2019. Using model theory in phonology: A novel characterization of syllable structure and syllabification. Doctoral dissertation, University of Delaware.

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*, vol. 3, chap. 7. Springer.

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.

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.

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.

Yip, Moira. 2002. *Tone*. Cambridge University Press.