Computational Phonology

Jane Chandlee, Haverford College and Jeffrey Heinz, University of Delaware

https://doi.org/10.1093/acrefore/9780199384655.013.116
Published online: 29 March 2017

Summary

Computational phonology studies the nature of the computations necessary and sufficient for characterizing phonological knowledge. As a field it is informed by the theories of computation and phonology.

The computational nature of phonological knowledge is important because at a fundamental level it is about the psychological nature of memory as it pertains to phonological knowledge. Different types of phonological knowledge can be characterized as computational problems, and the solutions to these problems reveal their computational nature. In contrast to syntactic knowledge, there is clear evidence that phonological knowledge is computationally bounded to the so-called *regular* classes of sets and relations. These classes have multiple mathematical characterizations in terms of logic, automata, and algebra with significant implications for the nature of memory. In fact, there is evidence that phonological knowledge is bounded by particular *subregular* classes, with more restrictive logical, automata-theoretic, and algebraic characterizations, and thus by weaker models of memory.

Keywords: phonology, phonotactics, alternations, computational linguistics, finite-state automata, finite-state transducers, monadic-second order logic, subregular hierarchy, learning algorithms and models

Subjects: Computational Linguistics, Linguistic Theories, Phonetics/Phonology, Psycholinguistics, Syntax

1. Phonology and Computation

Phonology is a theory that characterizes the knowledge people have regarding the way morphemes, words, and phrases are pronounced in their language. Computational phonology studies the nature of the necessary and sufficient computations entailed by such knowledge.

Computationally, phonological knowledge is a *problem*. A problem can be thought of as a *function* that takes instances of the problem as input and generates answers to the problem as output. As an example, consider the knowledge of how to sort lists of numbers in increasing order. As a problem, this is expressed as a function from its instances (lists of numbers) to their solutions (a sorted list). For example, an instance of this sorting problem is [20, 6, 18, 5] and its answer is [5, 6, 18, 20]. We define an *algorithm* as a procedure that *correctly* supplies

an answer to problem when given *any* instance of it. A problem is considered *solved* if there exists an algorithm that for it. Different problems may require different kinds of computations of the algorithms that solve them.

Three classes of important problems in phonology are discussed from this perspective: membership problems, translation problems, and learning problems. These problems correspond to three critical aspects of the nature of phonological knowledge: phonotactic knowledge, knowledge of phonological transformations, and the fact that this knowledge is acquired through experience. These problems are discussed primarily in terms of string representations. (Here, a string is a finite sequence of symbols. Following traditional notation, the set of all logically possible strings constructed from a finite set of symbols Σ is denoted Σ^*.) Issues involving alternative representations are also discussed. Other articles discussing computational phonology are Heinz (2011a, 2011b) and Daland (2014).

2. Phonotactic Knowledge and the Membership Problem

Phonotactic knowledge is the knowledge speakers have regarding the well-formedness of possible words in their language. For example, *gdark* is not a possible word in English but *blark* is a possible word. Phonotactic knowledge is language-specific: *gdark*, for example, is a possible Polish word.

In classical terms, the membership problem can be stated for any two sets A, B such that A is a subset of B (written $A \subseteq B$). The problem is to determine which elements of B are members of A. So every element x of B is an instance of the membership problem and its answer is 1 iff x is in A and 0 otherwise. The membership problem may also be considered in non-classical ways. For instance, one non-classical interpretation is in terms of fuzzy membership. In this case, the answers to instances of the membership problem may be taken to be points in the real interval [0,1]. These values can be interpreted as 'degrees of membership'. (They may also be interpreted as probability values.)

Classically, phonotactic knowledge of a particular language is thus a membership problem $M : \Sigma^* \rightarrow \{0,1\}$. (Note: we write $f : A \rightarrow B$ for a function f that maps elements of set A to elements of set B; A is called the *domain* and B the *co-domain* of f.) Out of the logically possible strings, which strings are well-formed (and therefore are 'members' of the set of possible words in the language)? Many phonologists today—but not all—prefer to view phonotactic knowledge in terms of fuzzy membership $M : \Sigma^* \rightarrow [0,1]$ (Coleman & Pierrehumbert, 1997; Hayes & Wilson, 2008; Daland et al., 2011; Gorman, 2013), in which case the question becomes: out of all the logically possible strings, to what degree are individual strings well-formed?

Both the classical and fuzzy membership problems for phonotactic knowledge have the same instance space: Σ^*. Formal language theory studies the nature of the necessary and sufficient computations that algorithms must make when solving membership problems with this

instance space. The classical membership problem is focused on because the computational nature of the problem changes little when the fuzzy membership problem is considered instead.

Two additional related problems help illustrate how different problems are similar in terms of their computational nature. One—familiar to contemporary phonologists—is determining the number of times a string violates a constraint. Letting $\mathbb{N}$ refer to the set of non-negative integers, this problem is a function $M : \Sigma^* \rightarrow \mathbb{N}$, which maps strings to numbers. For any constraint posited in phonology, we are interested in algorithms that solve this 'problem'; that is, algorithms that correctly compute the number of violations for each string in the instance space of the problem. Another problem is $M : \Sigma^* \rightarrow \Delta^*$, where Δ is another finite set of symbols of (not necessarily disjoint from Σ). This is the 'translation problem'.

The fuzzy membership problem and the others mentioned here are significantly informed by the nature of the classical membership problem. This is because if the instance space of the problem is Σ^* and if the answer space has the properties of a *semiring*, then there is a level at which all of these problems can be analyzed in the same way (Goodman, 1999).

A semiring is a set of values equipped with two operations with certain properties (a formal definition is given Figure 1). By convention these operations are called 'addition' and 'multiplication', though their actual interpretation will depend on the nature of the values of the semiring. The simplest example of a semiring is the set of natural numbers, in which case addition and multiplication have their familiar interpretations. In this example, 0 is the *additive identity*, since adding 0 to any integer gives the integer itself, and likewise 1 is the *multiplicative identity*. This is the semiring employed in Harmonic Grammar (HG) (Potts, Pater, Bhatt, & Becker, 2008) when constraint weights are required to be nonnegative integers.

Figure 2 presents three additional semirings. The answer space for the classical problem is the Boolean semiring. This is the set {true, false}, with the addition operator being OR ($\vee$) and the multiplication operator being AND ($\wedge$). The Boolean semiring is one of the simplest semiring structures, which makes it convenient to use it for exposition as we do here. Another semiring is the Viterbi semiring, which is often used in natural language processing tasks (Jurafsky & Martin, 2008). The Language semiring has not been studied much, but it is a natural choice for studying transformations.

A *semiring* is a set K over which two binary operations are defined $\oplus, \otimes$, called 'addition/plus' and 'multiplication/times' which contains elements 1 and 0 with the following properties satisfied: if $x, y, z \in K$ then

- $x \oplus y, x \otimes y \in K$ (closure under $\oplus$ and $\otimes$)
- $x \oplus y = y \oplus x$ ($\oplus$ is commutative)
- $0 \oplus x = x \oplus 0 = x$ (0 is the identity for $\oplus$)
- $1 \otimes x = x \otimes 1 = x$ (1 is the identity for $\otimes$)
- $0 \otimes x = x \otimes 0 = 0$ (0 is an annihilator for $\otimes$)
- $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$. ($\otimes$ right distributes over $\oplus$)

Figure 1. Definition of a semiring.

Name	K	$\oplus$	$\otimes$	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	$\mathcal{P}(\Sigma^*)$	$\cup$	$\cdot$	$\emptyset$	$\{\lambda\}$

Figure 2. Some semirings.

Many computational results transfer directly to the other cases when the answer space changes from one semiring to another. This is because the abstract structure of all semirings is responsible for the results, not the particulars of the semiring itself.

2.1. Formal Language Theory

A cornerstone of formal language theory is the Chomsky Hierarchy (see Figure 3), which divides subsets of Σ^* into nested regions: Finite, Regular, Context-Free, Context-Sensitive, and Computably Enumerable. Each of these terms corresponds to a level of computational power, with Finite computations being the weakest and Computably Enumerable computations being the strongest. The nesting of the levels entails that subsets in the Finite region are also in the Regular region, subsets in the Regular region are also in the Context-Free region, and so on.

Every region in the Chomsky Hierarchy can be defined in multiple ways. Each of these ways highlights different features of the different algorithms that may be employed to solve the membership problem for languages in those regions. However, the fact that these different definitions converge to the same regions shows that they have something deep in common.

Ultimately, it may be said that these regions correspond to different types of memory models. In order to solve the membership problem, an algorithm must be able to remember certain features of each input string as it is processed. When viewed in this light, these regions have immediate interpretations in psychology and cognition.

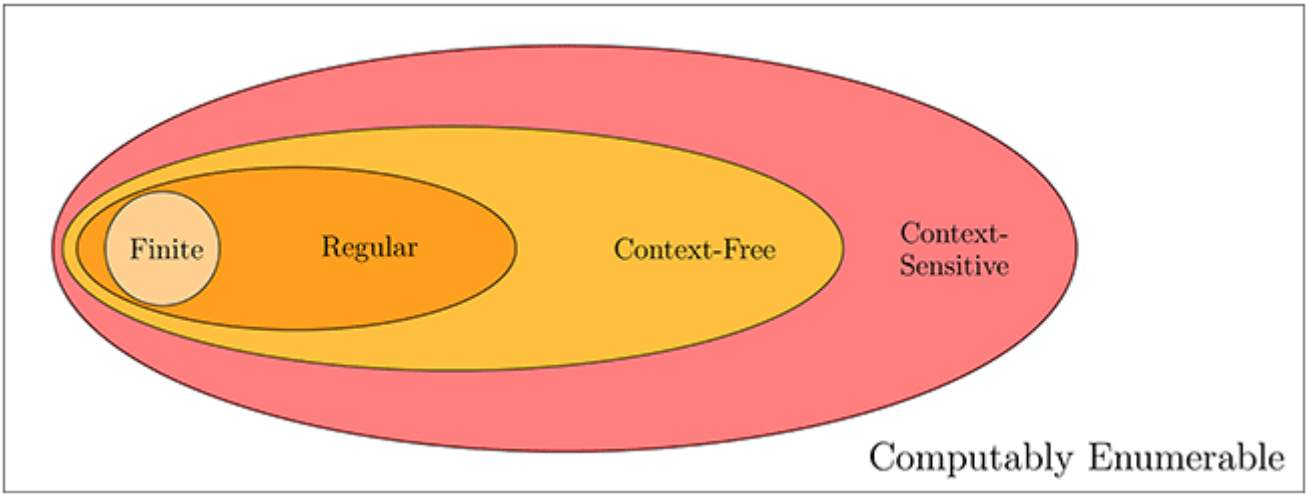


Figure 3. The Chomsky Hierarchy. Points in this space are subsets of Σ^*.

Statements like “Phonotactic Knowledge is Regular but not Finite” is an example of how formal language theory provides a way to bound the nature of the computations entailed by phonotactic knowledge both from above and from below. More detail will be provided about what it means for knowledge to ‘be finite’ and ‘be regular’. What is at stake for such hypotheses in terms of the insights formal language theory can offer is briefly discussed.

For a phonotactic constraint to be a regular language means there is a fixed, finite bound on the amount of information any computational device (including a speaker/parser, for example) needs to remember to solve the membership problem. For example, consider Navajo sibilant harmony (Sapir & Hojier, 1967), in which suffix sibilants must agree with sibilants in the root in anteriority. A parser has the capacity to determine whether strings of arbitrary length obey the constraint, and to do so it only needs to scan the string left to right and determine which of the following statements along the way: (1) no sibilant has been found, (2) a [+anterior] sibilant was found first, or (3) a [− anterior] sibilant was found first. If statement (1) is true and the scanner reaches the end of the string, the string is in the language. If statement (2) or (3) is true and the scanner finds a [− anterior] sibilant or a [+anterior] sibilant, respectively, then the string is not in the language. As will become clear later, these disjoint statements that the scanner keeps track of correspond to the ‘states’ of any finite state machine (see § 2.3) that describes this pattern.

In contrast, a pattern that cannot be represented with a fixed finite set of states is non-regular. As an example, consider a language in which all words must be palindromes (i.e., in a word $x_1 x_2 \ldots x_{n-1} x_n$ with n segments, $x_1 = x_n$, $x_2 = x_{n-1}$ as with *masakasam*). Any parser in this case must remember every segment encountered until the end of the word and then test

for the needed equalities. Since the length of words is unbounded (see § 2.2), the memory requirements are also unbounded in the relevant way (specifically, not finite state). Note also that while sibilant harmony is well-attested in natural language, the 'palindrome' phonotactic constraint is both unattested and intuitively non-phonological.

These two examples show how formal language theory draws a clear and well-defined boundary between an attested, regular phonotactic pattern and an unattested, non-regular one. Furthermore, the classification reflects computational memory requirements that have a clear interpretation for cognitive load and processing. In sum, formal language theory is a framework that leads to restrictive and testable predictions about what is possible in natural languages.

2.2. The Finite Region

The finite region includes all and only those subsets of Σ^* with finitely many strings. If $P \subseteq \Sigma^*$ has finite cardinality, there is a simple algorithm that can solve the classical membership problem for P. The algorithm contains a table that lists every element of P. For every instance x of the problem, the algorithm checks whether x occurs in this table. If it does, it returns 1; if not, it returns 0.

With this in mind, we can ask: is the phonotactic knowledge of every human language finite? One reason to think that phonotactic knowledge is finite is because there are only finitely many words in any human's mental lexicon at any one time. However, humans can also coin new words (like *blark*), which shows that the number of *possible* words in the language always exceeds the number of actual words.

The answer to the prior question hinges on whether there is an upper bound on the length of words. If there are finitely many possible well-formed words then there is an upper bound on the length of words (this bound will be one more than the longest word). Conversely, an upper bound on the length of words entails there are only finitely many well-formed ones. Many scholars believe there is no upper bound on the length of possible words in languages. For example, here is a long nonce word of English that we think is well-formed: *kàpalàsakòulapinìpisàukimàlagàlanú*. Given any positive integer n, we believe we can construct a possible word of English of length larger than n (see also Daland, 2015). (We recognize that for large n these words are never fully pronounceable because human lives are not long enough to utter them.)

One reason to think the answer is No has an element of practicality to it. Savitch (1993) argues that even if there are only finitely many words, in some cases it is better to treat the language as infinite anyway. This is because large finite sets of strings can often be factored into two parts: an infinite set of strings and a separate finite-length condition.

Here is a concrete example illustrating the point. Consider the set $S = \left\{a, aa, aaa, \ldots a^{1000}\right\}$. Here a^n means the symbol a is repeated n times. S is a finite set of strings. As stated, an algorithm can solve the membership problem for this set by containing a table with 1,000

entries. Given any string $x \in \Sigma^*$, it can check this table entry by entry to see whether x occurs in this list. Of course there is another alternative. For any string, an algorithm can simply check whether every symbol in the string is *a* and also check that the length of the string is between one and 1,000, inclusive. This latter algorithm also solves the membership problem for S, but does so by factoring out the length requirement from the question of whether every symbol in the input string is *a*. The question of whether every symbol in any string in Σ^* is *a* is a problem that does *not* fall in the finite region. Solving this problem *cannot* be done with any finite list of strings. Instead it requires a different kind of computational power (regular power; see § 2.3).

This example illustrates Savitch's point that treating finite sets as essentially infinite has benefits. The program that checks whether every symbol in a string is *a* requires substantially less memory than storing a table with 1,000 entries. On the other hand, it requires a (slightly) more complex procedure than one that checks entries in a table to see if one matches the instance of the problem.

2.3. The Regular Region

2.3.1. Overview

There are several kinds of algorithms that can solve the membership problem for sets of strings in the regular region. These algorithms rely on different formal expressions that specify regular subsets of Σ^*. They can be specified in many ways, including three well-studied ones: in terms of *Monadic Second-Order (MSO) logic*, in terms of *finite-state acceptors*, and in terms of *regular expressions*. For each type of grammar, there is an effective procedure (algorithm) that takes as input a grammar in that formalism and a string in Σ^* and correctly outputs the solution to the membership problem defined by that string and grammar.

For any regular set P, MSO logic has the advantage that it is a high-level specification language (not unlike a programming language such as Python or Haskell). This allows one to state P in an unambiguous, readable manner. On the other hand, finite-state acceptors are a low-level language and so it is easy to define procedures that operate on automata (not unlike low-level computer languages like assembler languages). For this reason, the most widely used algorithms that solve the membership problem are based on finite-state acceptors. However, finite-state acceptors can be difficult to read, especially if they are not small. Regular expressions can also be difficult to read if they are large, and so many convenient abbreviations have been introduced for them and are in wide use. Unlike MSO logic and finite-state acceptors, regular expressions are most commonly employed for the classical membership problem and only for problems whose answer space is the Boolean semiring (see Figure 2; an important exception is discussed in §3).

These three formalisms will be introduced informally with examples. Readers are referred to Hopcroft, Motwani, and Ullman (2001) and Enderton (2001) for formal definitions. The first example (Figure 4) shows each formalism for the problem of determining whether a string

contains only a. The second example (Figure 5) shows each formalism for the problem of determining whether a string of stressed and unstressed syllables is well-formed in Maranungku.

2.3.2. Example: Only-a

Consider the three grammars in Figure 4, each of which describes the set of strings $\{a, aa, aaa, \ldots\}$.

MSO formulae. MSO formulae contain variables like x, which range over positions in the string. Positions are standardly represented with numbers. For example string abc has three positions {1,2,3}. MSO formulae are constructed recursively from primitive formulae. The primitive formulae for strings typically include predicates for how the positions are labeled. For example, the primitive formulae $\mathrm{a}(x)$, $\mathrm{b}(x)$, $\mathrm{c}(x)$ each evaluate to true for the string abc when x takes on the values 1, 2, and 3, respectively. So the above MSO expression says "For all positions x, x is a and there is a position x such that x is a." (This particular example is actually an example of First-Order (FO) logic, which is a fragment of MSO logic.)

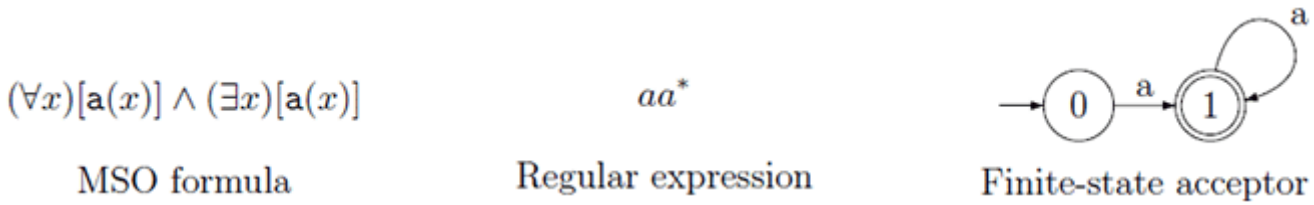


Figure 4. Three different descriptions of the same set of strings $\{a, aa, aaa, \ldots\}$.

Regular Expressions. In the previous regular expression, the Kleene star (*) operation means "zero or more occurrences" so a^* means zero or more a's in sequence, and aa^* means a string with one a followed by zero or more a's.

Finite-state Acceptors. Finite-state acceptors are a kind of graph. The labeled arrows in the graph are called *transitions* and the circles of the graph are called *states*. The transitions that occur among states define a set of *paths*. A word w satisfies the acceptor provided there is a path that begins at the start state (here indicated by the incoming arrow in state 0), ends at a final state (here indicated by the double circle of state 1), and whose concatenation yields w. Here is the path for aaa: $0 \xrightarrow{a} 1 \xrightarrow{a} 1 \xrightarrow{a} 1$. Observe there is no path for $aaba$.

2.3.3. Example: Maranungku Stress

Maranungku places primary stress on the first syllable and secondary stress on subsequent alternating syllables (Halle & Vergnaud, 1987). For instance, the following strings of syllables are well-formed. (Maranungku actually has a prohibition against monosyllabic words; we do not treat this fact here.)

σ́	σ́σ̀	σ́σ̀σ̀	σ́σ̀σ̀σ̀
σ́σ	σ́σ̀σ	σ́σ̀σ̀σ	σ́σ̀σ̀σ̀σ

Figure 5 shows three descriptions of the linguistic generalization that these forms are examples of: one in terms of MSO logic, one in terms of a regular expression, and one in terms of a finite-state acceptor.

MSO formulae. Here, the grammar given by MSO formula defines several constraints and Maranungku is simply the conjunction of four constraints: the first ($\rtimes\acute{\sigma}$) requires that the first syllable bear primary stress, and the others forbid primary stress on any syllable but the first ($\neg\mathtt{x}\acute{\sigma}$), lapses ($\neg$lapse) and clashes ($\neg$clash). The auxiliary formulae first(x) is only true for positions x for which there is no position y where x is the successor of y (written '$y \lhd x$'). The successor relation is one of the atomic formulae in MSO logic with successor. In general, atomic formulae correspond to the representational primitives.

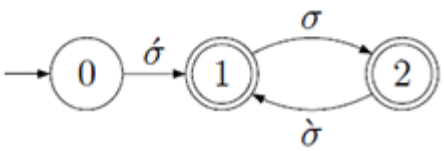
$$
\begin{aligned}
\mathtt{first}(x) &\stackrel{\text{def}}{=} \neg(\exists y)[y \lhd x] \\
\mathtt{stress}(x) &\stackrel{\text{def}}{=} \acute{\sigma}(x) \vee \grave{\sigma}(x) \\
\rtimes\acute{\sigma} &\stackrel{\text{def}}{=} (\exists x)[\mathtt{first}(x) \wedge \acute{\sigma}(x)] \\
\mathtt{x}\acute{\sigma} &\stackrel{\text{def}}{=} (\exists x)[\acute{\sigma}(x) \wedge \neg\mathtt{first}(x)] \\
\mathtt{lapse} &\stackrel{\text{def}}{=} (\exists x, y)[x \lhd y \wedge \sigma(x) \wedge \sigma(y)] \\
\mathtt{clash} &\stackrel{\text{def}}{=} (\exists x, y)[x \lhd y \wedge \mathtt{stress}(x) \wedge \mathtt{stress}(y)]
\end{aligned}
$$

$$\mathtt{Maranungku} \stackrel{\text{def}}{=} \rtimes\acute{\sigma} \wedge \neg\mathtt{x}\acute{\sigma} \wedge \neg\mathtt{lapse} \wedge \neg\mathtt{clash}$$

MSO formula

$\acute{\sigma}(\sigma\grave{\sigma})^* + \acute{\sigma}(\sigma\grave{\sigma})^*\sigma$

Regular expression



Finite-state acceptor

Figure 5. Three descriptions of Maranungku stress.

Regular expressions. The regular expression for Maranungku uses the '+' (plus) operator, which denotes the union of the denotations of the two expressions it combines. The denotation of the expression on the left is all strings that begin with $\acute{\sigma}$ followed by zero or more occurrences of the string $\sigma\grave{\sigma}$. The denotation of the expression on the left is the same as the one on the right except that the strings must end with σ.

Finite-state acceptors. The finite-state acceptor for Maranungku has two final states, corresponding to whether the string ends with a σ or $\grave{\sigma}$. Here is the path for $\acute{\sigma}\sigma\grave{\sigma}\sigma\grave{\sigma}$:
$0 \overset{\acute{\sigma}}{\to} 1 \overset{\sigma}{\to} 2 \overset{\grave{\sigma}}{\to} 1 \overset{\sigma}{\to} 2 \overset{\grave{\sigma}}{\to} 1$.

2.3.4. Additional Important Facts

The equivalence of regular expressions and finite-state acceptors was determined by Kleene (1956). The equivalence of finite-state acceptors and MSO logic (with successor, see §2.4) was determined by Büchi (1960).

Of the three types of descriptions, the MSO logical formulae may appear to be most similar to traditional linguistic theory at first glance. This is because new expressions can be compositionally combined from more primitive ones to define constraints recognizable from the phonological literature. For example, in Maranungku (Figure 5) constraints akin to *CLASH and *LAPSE (Tesar & Smolensky, 1998) were defined. However, regular expressions also build up complex expressions from simpler ones. In terms of regular expressions, LAPSE could be defined as all and only those strings containing a lapse.

$$\text{lapse} \overset{\text{def}}{=} (\acute{\sigma} + \grave{\sigma} + \sigma)^{*}\, \sigma\sigma\, (\acute{\sigma} + \grave{\sigma} + \sigma)^{*}$$

Generalized regular expressions extend regular expressions by adding symbols for relative complement and intersection (McNaughton & Papert, 1971). This notation allows one to easily express all the words that do *not* contain a lapse. This constraint can then be intersected with the sets of strings corresponding to those that do not contain a clash, and so on. Despite the extra symbols and their interpretations, generalized regular expressions are equivalent to regular expressions; that is, both define the same sets of strings—the regular stringsets.

MSO formulae naturally allow 'weights' to be added by changing the nature of the semiring (Figure 2). When using the Natural semiring instead of the Boolean semiring, MSO logical formulae provide a constraint definition language in the sense of de Lacy (2011), which can count the number of violations.

$$\text{LAPSE} \overset{\text{def}}{=} (\exists x, y)[x \lhd y \wedge \sigma(x) \wedge \sigma(y) \wedge 1]$$

The ‘$\wedge 1$’ has the effect of assigning one to each (x, y) pair that satisfies the existential clause. Then the existential quantifier is interpreted as the sum of these values. If we wanted to define lapse so that each violation is counted twice we would only need to replace ‘$\wedge 1$’ with ‘$\wedge 2$’. See Droste and Gastin (2009) for details.

On the other hand, we are unaware of a general solution to weighting regular expressions, though there has been considerable work on extending regular expressions for both the membership and translation problems (Beesley & Karttunen, 2003; Hulden, 2009a), as discussed in §5.4.

Finite-state acceptors are just one kind of finite-state machine. Finite-state machines (also called finite-state automata) come in several varieties: they can be either one-way or two-way, they can read strings either left-to-right or right-to-left, they can be either deterministic or non-deterministic, and in addition to input symbols on the transitions, the transitions may include output labels (Savitch, 1982; Sipser, 1997). Typically, the output labels include elements from a semiring such as natural numbers, real values, or strings (Eisner, 2003). The automata shown in Figures 4 and 5 are one-way, deterministic finite-state machines without output labels that read strings left-to-right. (In general, the term ‘acceptor’ is reserved for finite-state machines without output labels on the transitions.) When the transitions do not include output labels, all of the varieties are equally expressive. For instance, any membership problem solvable with two-way, non-deterministic finite-state acceptors are also solvable with one-way, deterministic finite-state acceptors, and vice versa (Savitch, 1982).

2.4. The Subregular Hierarchy

It is natural to ask whether better bounds on the computational nature of phonotactic knowledge can be obtained. The Subregular Hierarchy is shown in Figure 6 (McNaughton & Papert, 1971; Rogers & Pullum, 2011; Rogers et al., 2013). Like the classes in the Chomsky Hierarchy, these classes are nested and are interpretable in terms of memory load and processing requirements. If a line connects region A to region B in the diagram, and region A is higher than region B, then region A properly contains region B. The Subregular Hierarchy divides the regular regions along two dimensions: choice of representational primitives and logical power. The logical dimension is split into four types of logic in decreasing power: MSO logic, First-Order (FO) logic, Propositional logic, and Conjunctions of Negative Literals (CNL), which is a fragment of propositional logic. The representational dimension is split into the two ways linear order can be represented: immediate successor ($\lhd$) and general precedence ($<$). The distinction between these representations of order can be illustrated with a word like bat. It has two successor relations: $b \lhd a$ and $a \lhd t$, but three precedence relations: $b < a$, $b < t$ and $a < t$.

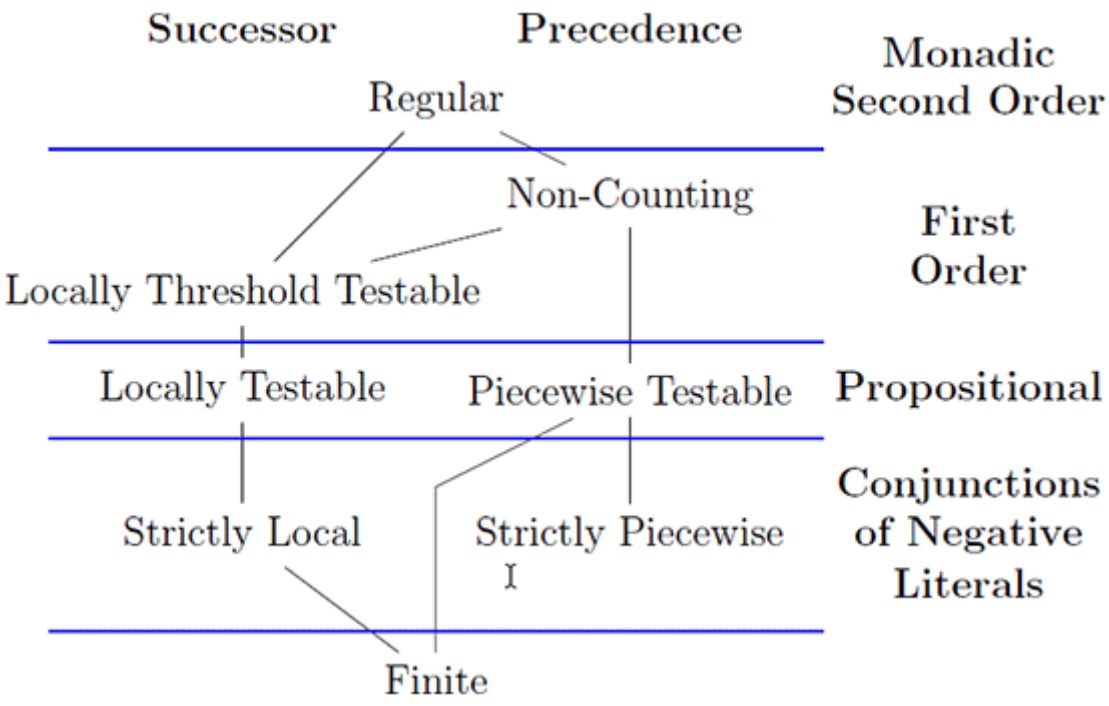


Figure 6. The Subregular Hierarchy.

As explained by Rogers et al. (2013), each of the classes in the Subregular Hierarchy corresponds to a particular algorithmic model of memory and perception. For instance, the CNL classes correspond to the simplest, non-finite memory models. These algorithms scan words in terms of sub-structures (of some fixed size) as determined by the representational primitives. On the successor side, these sub-structures are substrings (for instance ba, at, and ts are substrings of length two in $bats$), but on the precedence side, these sub-structures are subsequences (for instance ba, bt, bs, at, as, and ts are subsequences of length 2 in $bats$). Thus the Strictly Local class contains exactly those membership problems that can be solved by algorithms that check the substrings in words against a finite list of forbidden substrings. Similarly, the Strictly Piecewise class contains exactly those membership problems that can be solved by algorithms that check the subsequences in words against a finite list of forbidden subsequences.

The membership problems in the higher classes can be solved by algorithms with more elaborate, enriched memories. For instance the Locally Testable and Piecewise Testable classes are the Boolean closure of the Strictly Local and Strictly Piecewise classes, respectively. Algorithms solving membership problems in these regions must not only check which sub-structures of fixed size are present in words but must also run the results of these checks through a Boolean circuit to correctly decide whether the answer to a given instance of the problem is 0 or 1.

The classes in the Subregular Hierarchy provide a striking example of the interplay between representation and algorithmic power. The analysis of certain types of long-distance phonotactic patterns depends critically on how linear order is represented. To explain, consider one type of phonotactic pattern found in natural language: the one derived from a

productive process of asymmetric sibilant harmony like in Sarcee (Cook, 1978, 1984). In this language there are no words with a non-anterior sibilant (like [ʃ]) following an anterior sibilant (like [s]). The examples below (from Cook, 1978) illustrate this generalization in Sarcee: anterior sibilants like [s] may occur after, but not before, non-anterior sibilants like [ʃ].

/na-s-ɣat ʃ/	→	nā ʃ ɣát ʃ	'I killed them again'
/si-t ʃ iz-aʔ/	→	ʃ ít ʃ ídzàʔ	'my duck'(cf. *sít ʃ ídzàʔ)

This generalization is conventionally represented with notation like *s . . . ʃ (Hansson, 2010). If linear order is represented with successor (◁) then full MSO logical power is needed to solve the membership problem for this pattern as shown in Figure 7. (In the following formalizations and in the figures, we represent [ʃ] with [S].) The formula sarcee defines general precedence in terms of successor. General precedence is the transitive closure of successor. This is expressed in the formula closed(X), which takes a monadic second-order variable X that ranges over subsets of the domain. The necessity of this second-order variable is what makes MSO power necessary for defining precedence from successor. On the other hand, if linear order is represented with precedence (<), the same membership problem can be solved merely with the conjunction of negative literals! Now the formula is simply $\neg sS$.

$$
\begin{aligned}
\mathtt{s}(x) &\stackrel{\text{def}}{=} \mathtt{anterior}(x) \wedge \mathtt{sibilant}(x) \\
\mathtt{S}(x) &\stackrel{\text{def}}{=} \neg\mathtt{anterior}(x) \wedge \mathtt{sibilant}(x) \\
\mathtt{closed}(X) &\stackrel{\text{def}}{=} (\forall x, y)\big[(x \in X \wedge x \lhd y) \Rightarrow y \in X\big] \\
x < y &\stackrel{\text{def}}{=} (\forall X)\big[(x \in X \wedge \mathtt{closed}(X) \Rightarrow y \in X\big] \\
\mathtt{Sarcee} &\stackrel{\text{def}}{=} \neg(\exists x, y)[x < y \wedge \mathtt{s}(x) \wedge \mathtt{S}(y)]
\end{aligned}
$$

Figure 7. A MSO(◁) formula for Sarcee sibilant phonotactics.

This example shows that the difference in terms of memory requirements between local and long-distance phonotactics is qualitative because it depends on the representation.

For completeness, Figure 8 presents a generalized regular expression and finite-state acceptor description for the same Sarcee phonotactics. In the finite-state acceptor, 'x' is an abbreviation for any non-sibilant.

Heinz (2010a) hypothesizes that all phonotactic knowledge is Strictly Local (i.e., based on successor) or Strictly Piecewise (i.e., based on precedence). The appeal of this hypothesis is that, as indicated in Figure 6, the computations needed to solve the membership problem in phonology would be on the most restrictive level of the hierarchy (aside from Finite). However, long-distance patterns with blocking are a problem for this hypothesis (see Heinz, 2010a for details).

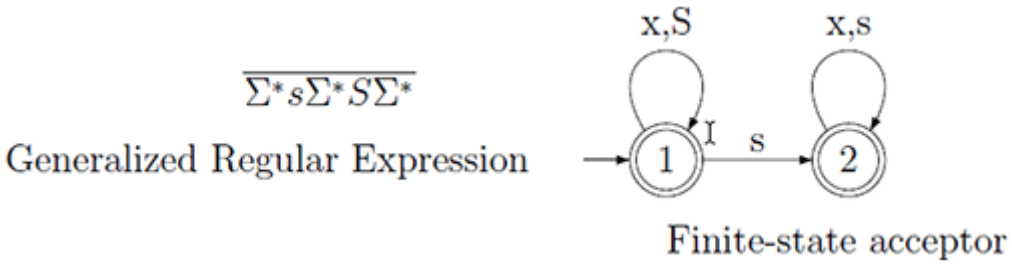


Figure 8. A generalized regular expression and finite-state acceptor for Sarcee sibilant phonotactics.

An alternative hypothesis is that phonotactic knowledge is Tier-based Strictly Local (Heinz, Rawal, & Tanner, 2011), a class of languages with the option of designating a subset of Σ as a *tier* over which Strictly Local constraints can then be defined. Similar to phonological analyses that assume autosegmental representations (Goldsmith, 1979), the Tier-based Strictly Local class allows one to capture dependencies among segments that are non-string-adjacent (i.e., these segments are only adjacent on the tier). The Tier-based Strictly Local class properly includes the Strictly Local class, which means local phonotactics describable with Strictly Local constraints are also describable with Tier-based Strictly Local constraints, and can also describe long-distance patterns including those with blocking. The Tier-based Strictly Local class is not shown in Figure 6, but it is properly included by the Non-Counting class.

One way to define the Non-Counting class is as the class of patterns that can be described with generalized regular expressions without the '*' notation (McNaughton & Papert, 1971). (Recall that an expression like a^* means a string of any number of a's). Since the Non-Counting class properly includes the Tier-based Strictly Local class, a weaker hypothesis is that phonotactic knowledge is Non-Counting. This hypothesis is weaker because—if indeed all phonotactic knowledge is Tier-based Strictly Local—then generalizing 'up' to Non-Counting fails to capture an important generalization regarding the nature of phonology.

Nonetheless, the hypothesis that phonotactic patterns fall no higher than Non-Counting on the hierarchy asserts that they are properly subregular. This hypothesis enjoys broad support and there are only two potential counterexamples to our knowledge: the predictable stress patterns of Creek and Cairene Arabic, which Graf (2010a) shows are more powerful than Non-Counting. The Cairene Arabic (Mitchell, 1960; Graf, 2010a) pattern is described as follows: stress the final syllable if it's superheavy, else stress the penultimate syllable if it's heavy, else stress whichever syllable (between the penultimate or antepenultimate) is separated from the

closest heavy syllable by an even number of syllables. Satisfying the third part of this description requires counting modulo 2, which the Non-Counting class is not capable of. Thus, this is an example of a pattern that is properly regular.

Whether the reported generalizations for Creek and Cairene Arabic are accurate merits further study (in particular whether secondary stress is perceptible is a crucial point that affects the pattern's computational status; see Graf, 2010a, for details). As mentioned, these are the only known counterexamples to the hypothesis that phonotactics are subregular. If subregularity was not a significant property of phonology one would expect to find many more such cases.

3. Morpho-phonological Alternations and the Translation Problem

Alternations in phonology refer to morphemes that have multiple pronunciations; the morphemes 'alternate' among them. How to predict the pronunciation of a morpheme in a particular context is one of the long-standing questions in theoretical linguistics. The hypothesis that forms the cornerstone of generative phonology is that humans store a single 'underlying representation' for each morpheme in their mental dictionary, and these underlying representations are transformed into the 'surface' variants (Kenstowicz & Kisseberth, 1979; Odden, 2014) according to language-specific conditions. Thus the alternations are explained in terms of a transformation of one representation into another.

These transformations are examples of the translation problem. The translation problem $T : \Sigma^* \rightarrow \Delta^*$ maps strings to strings (again, Δ is another alphabet of symbols, possibly distinct from Σ). The instances of the problem are possible underlying representations and the answers to the problem are the corresponding surface representations. For example, if we consider a phonology that only contains a process of word-final obstruent devoicing, then the following pairs are instances of the problem with their answers: (rat,rat), (rad,rat), (mab,map), (milo,milo).

What is the computational nature of phonological transformations? As with phonotactic knowledge, there is consensus that the statement "Phonological transformations are not Finite but Regular" is true. Before we present the evidence in favor of this statement, let us make clear the claim itself.

3.1. Types of Transducers

Within formal language theory, the computational landscape for translation problems is different from the one for membership problems, and this difference can be seen when comparing the finite-state representations used in the two problems. For the membership problem we used finite-state acceptors; for the translation problem we use finite state *transducers*. The distinction between these two types of finite-state machine is that the transitions in transducers have both input and output labels. These output labels are strings from Δ^* (or the set of all subsets of Δ^*, to accommodate variation/optionality).

Figure 9 shows different subclasses of the translation problem in terms of the kinds of transducers necessary to generate them. Like the classes in the Chomsky and Subregular Hierarchies (Figures 3 and 6), these classes are nested.

Unlike the case with finite-state acceptors, where the 1-way/2-way, deterministic/non-deterministic, left-to-right/right-to-left distinctions make no difference with respect to expressivity, these distinctions do matter for finite-state transducers. These terms are defined informally.

Determinism vs. Non-determinism. A FST is deterministic if for each state q and for each symbol a in the input alphabet there is at most one transition from q labeled a. A non-deterministic FST is not held to this requirement.

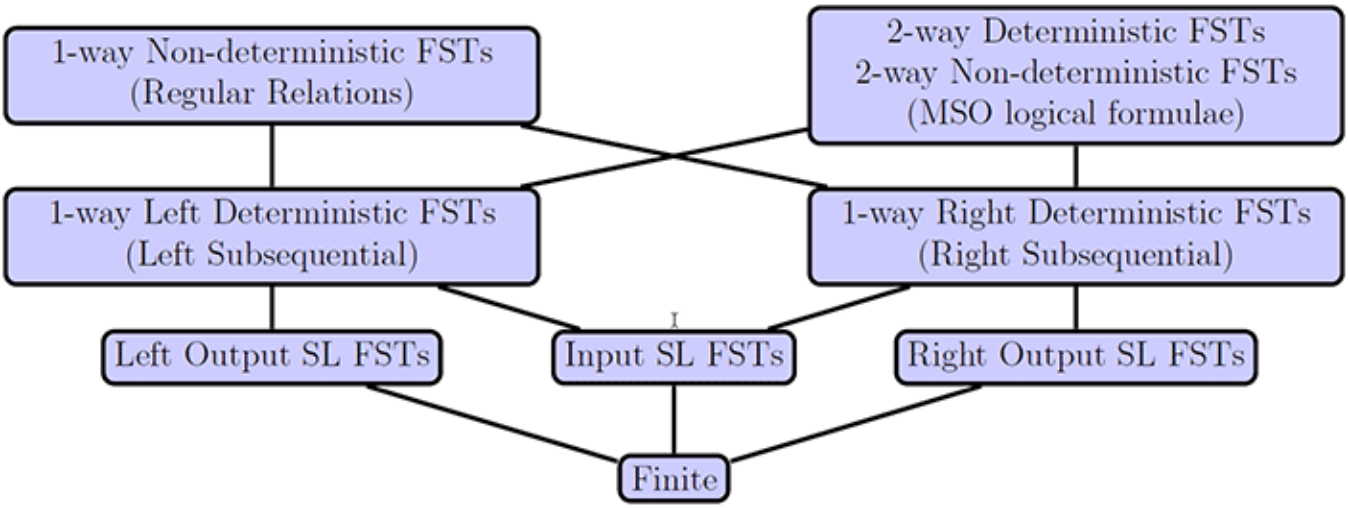


Figure 9. Subregular classes of transductions.

1-way vs. 2-way. A 2-way transducer is permitted to re-read parts of the input string. For instance, it may process the string once (and write some output) and then read it again (and write some more output). A 1-way transducer can read each part of the input string only once.

Left vs. Right. Left transducers process input strings left-to-right and build output strings with right-concatenation, whereas Right transducers process input strings from right-to-left and build output strings with left-concatenation. Right concatenation is the usual concatenation: $a \cdot_r b = \text{ab}$. Left concatenation is the reverse: $a \cdot_\ell b = \text{ba}$. To illustrate, consider the identity translation, which can be described with both Left and Right transducers and consider the input string *abcd*. The Left transducer reads as *a-b-c-d* and outputs $(((a \cdot_r b) \cdot_r c) \cdot_r d) = \text{abcd}$. The Right transducer would read it from the right as *d-c-b-a* and output $(((d \cdot_\ell c) \cdot_\ell b) \cdot_\ell a) = \text{abcd}$.

It is particularly striking that the 2-way deterministic string transductions are incomparable with 1-way non-deterministic transductions (so neither is a subset of the other). The former are exactly those transformations describable with MSO formulae (Engelfriet & Hoogeboom,

2001) and the latter are exactly the regular relations (Beesley & Karttunen, 2003). Recall that in the case of formal languages (Figure 6) MSO formulae corresponds exactly to regular. We see in Figure 9 that the situation is different for transductions. There are some non-regular transformations that can be described with MSO formulae. For example, total reduplication (where each input string w is mapped to ww) is non-regular (i.e., it cannot be described with a finite-state transducer), but this relation is definable with a MSO formula (in fact it can be defined with a First Order [FO] formula). Readers are referred to the recent survey by Filiot and Reynier (2016) for more details on these different types of transductions.

3.2. Why Phonological Transformations Are Regular Relations

As with the subregular hierarchy of formal languages, at the bottom of the hierarchy for transductions we again have the finite class. A finite transformation is only defined for a finite number of input strings. An algorithm can solve finite translation problems with a table with finitely many rows and two columns. It simply looks up the input in the first column and then writes out its corresponding output form (found in the second column). The reasons for not considering phonological transformations as finite are essentially the same as those previously given for the membership problem. We do not believe there is an upper bound on the length of underlying representations, and even if there were, it pays to treat them as essentially infinite.

Can phonological transformations be bounded from above? As mentioned, there is evidence that all phonological transformations are regular relations. Johnson (1972) and Kaplan and Kay (1994) show that phonological grammars that are defined as a list of rewrite rules like A $\rightarrow$B/C___D can be interpreted as regular relations, which as indicated in Figure 9 correspond to 1-way Non-deterministic FSTs. This is the first ingredient to the argument that phonological transformations are regular.

The second ingredient is composition. The list of rules $\langle R_1, R_2, \ldots R_n \rangle$ transform underlying forms to surface forms as follows: R_1 applies to the underlying form, then for each i greater than or equal to 2 but less than or equal to n, R_i applies to the output of R_{i-1}, and the surface form is the output of R_n. Regular relations are closed under composition (Scott & Rabin, 1959). This means that if R_1 and R_2 are regular relations, their composition $R_1 \circ R_2$ is also a regular relation. Hence, the entire grammar can also be expressed by a single FST. (The fact that a grammar of n rules can be represented by a single non-deterministic FST means the UR-SR mapping can be achieved without intermediate representations [Karttunen, 1993]; see also Bromberger and Halle [1989].)

The third ingredient is that the phonologies of the world’s languages appear amenable to descriptions as lists of ordered rules. These grammars may not be the most elegant or succinct, but they are sufficient to accurately describe the map from underlying forms to surface forms. In other words, we know of no phonology that cannot be described as a regular relation. However, can a stronger (i.e. more restrictive) characterization of the UR-SR mapping be given?

3.3. Subregular Classes of Transductions

Graf (2010b, 2010a) compares dependency-based theories of phonology such as Government Phonology (GP) (Kaye, Lowenstamm, & Vergnaud, 1990; van der Hulst, 2011, inter alia) with the Sound Pattern of English (SPE) (Chomsky & Halle, 1968). He finds that GP can be characterized by a modal logic with successor. Modal logic is generally recognized as being stronger than Propositional logic but strictly weaker than First Order Logic. He concludes that GP is more restrictive than SPE. He identifies a class of phonological patterns (spreading patterns) that cannot be captured by GP. He further concludes that GP as formulated is too restrictive unless it is extended in some ways he discusses. His discussion leaves open the question of whether there are proper subclasses of the regular relations that encompass all phonological transformations.

Other recent work seeks to answer this question by identifying *subclasses* of the regular relations to which phonological maps can be shown to belong. Several classes of transformations are represented in Figure 9, including the Input Strictly Local functions, Left and Right Output Strictly Local functions, and Left and Right Subsequential functions.

Input Strictly Local (ISL). Above the finite class are the Input Strictly Local FSTs (Chandlee, 2014; Chandlee & Heinz, forthcoming). ISL functions are those in which the output string associated with each input symbol depends only on the input symbol itself and the $k-1$ input symbols read. Figure 10 illustrates how the output of input symbol x is u, and u only depends on x and the $k-1$ input symbols preceding x (in this example $k=3$). Thus, Input Strictly Local FSTs (like all FSTs) read an input string left-to-right, one symbol at a time, and produce some portion of the output string at each step (i.e., the complete output string is the concatenation of the output strings produced by each transition). The ‘Input’ designation means only information in the underlying representation is used to determine what to output. The ‘Strictly Local’ designation means the output produced at each transition is only based on the most recent input, where ‘most recent’ is parameterized by a fixed integer k (see Figure 10). A k-Input Strictly Local FST thus has states for all possible sequences from Σ of length $k-1$, and the transitions are defined such that the FST is always in the state that matches the most recently read $k-1$ input symbols. In this way the FST keeps track of the most recent input and, importantly, nothing else. This restriction amounts to a short-term memory model, with the effect that only local, not global, properties of a string can be used in the transformation.

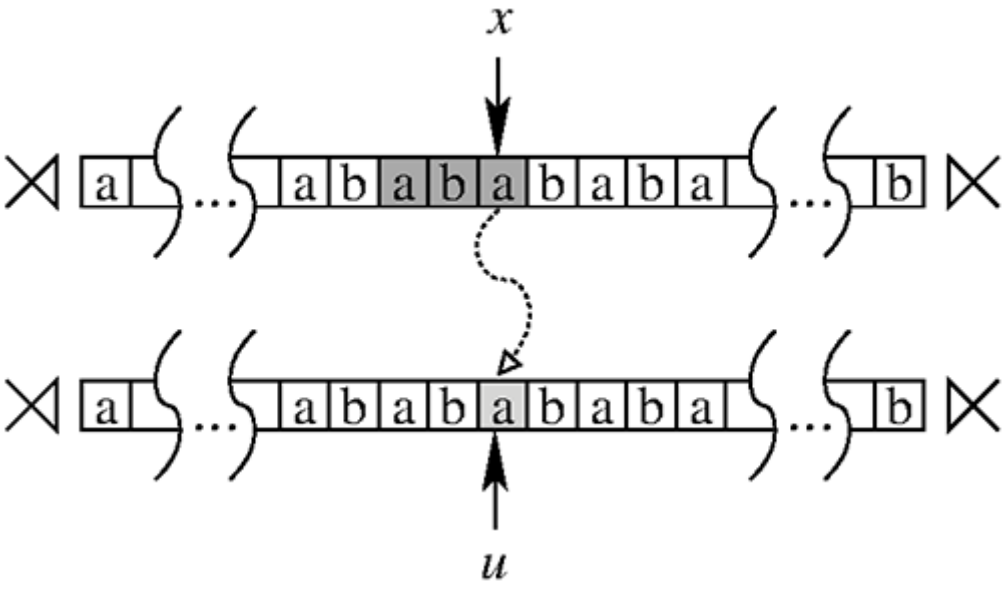


Figure 10. Input Strictly Local functions.

An example of a transformation that can be described with such a FST is post-nasal obstruent voicing (Pater, 2004), in which a voiceless obstruent that follows a nasal surfaces in its voiced form. Figure 11 presents the Input Strictly Local FST for this process, for which $k = 2$. For readability, a reduced alphabet is used, such that N = nasal, T = voiceless obstruent, D = voiced obstruent, and V = vowel.

The start state of the FST (shown in bold) is λ, which represents the empty string (i.e., the string with no symbols). Again, by definition the FST is always in the state that corresponds to the most recently read $k - 1$ input symbols, so it starts in a state that indicates it has not read anything. (Note: the '$:\lambda$' attached to each state label represents the value of the *final output function*, which appends additional output to strings that end in that state. In this example that function plays no role, because its value in all states is λ. See Chandlee, 2014 for more details.) The transitions are labeled with input and output (because this is a transducer and not an acceptor); a label such as N:N means an N is read from the input and an N is produced as output. The diagram makes clear that from any state, the N:N transition leads to state N (and likewise the other transitions lead to the respective T, D, and V states).

As an example, the path for the map of VNTV $\mapsto$ VNDV is shown.

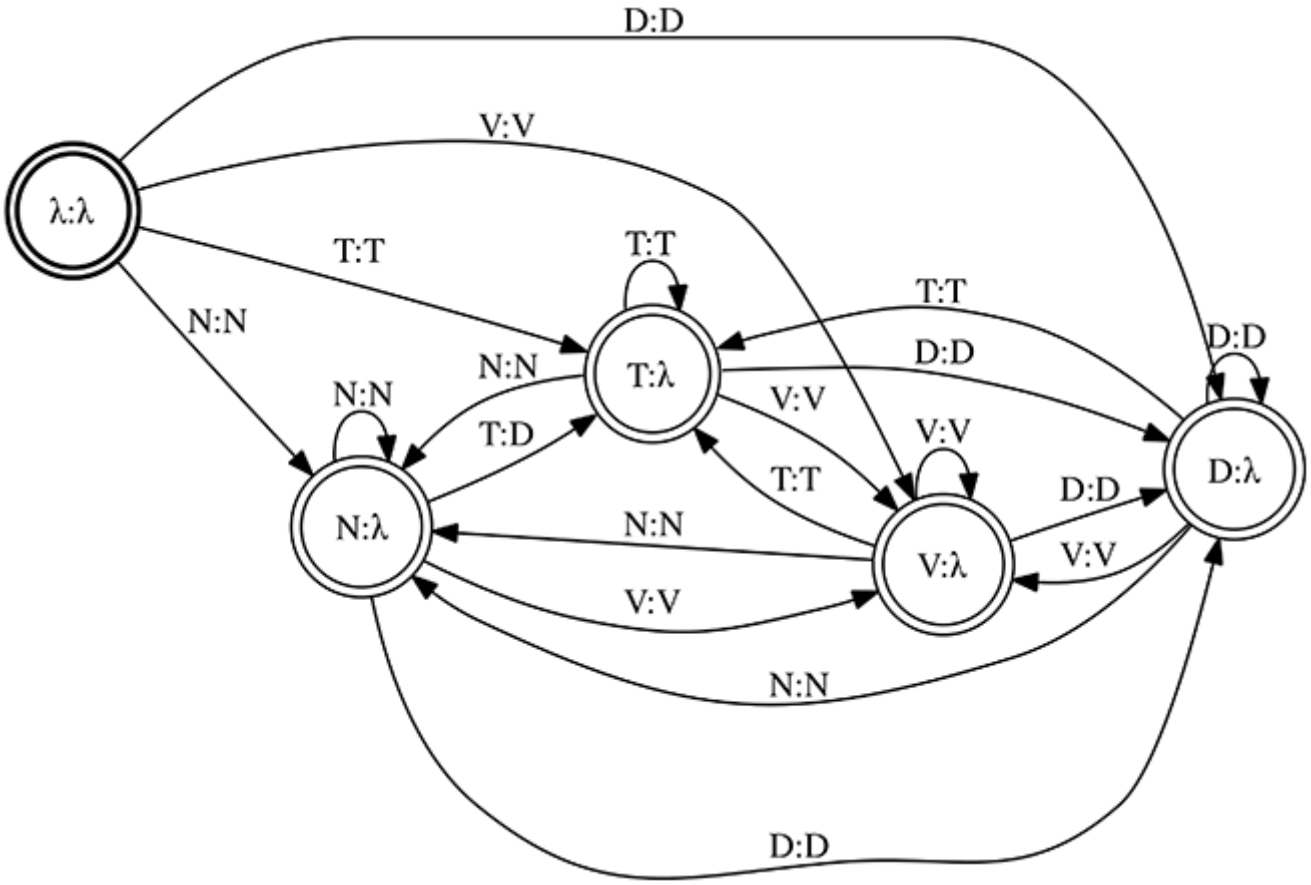


Figure 11. 2-ISL FST for post-nasal obstruent voicing.

Input:		V		N		T		V	
States:	λ	$\rightarrow$	V	$\rightarrow$	N	$\rightarrow$	T	$\rightarrow$	V
Output:		V		N		D		V	

Processes such as post-nasal obstruent voicing are traditionally categorized as 'local' because the target of the process is adjacent to the trigger. Input Strictly Local processes, however, also include those for which the target is a *bounded* number of segments away from the trigger. For example, in Ndonga nasal agreement (Viljoen, 1973; Rose & Walker, 2004), the liquid in the /-el/ suffix becomes nasal if the last segment in the stem it attaches to is a nasal (e.g., /kam-el-a/ $\mapsto$ [kamena], 'press for'). Here the trigger of the agreement is the stem-final nasal, but it is separated from the target by an intervening vowel. Importantly, the process does not occur if more than one vowel intervenes (i.e., the amount of intervening material is bounded by 1). This process has been classified as long-distance because it involves non-adjacent segments (Rose & Walker, 2004; Hansson, 2010), but the existence of a bound on the intervening material gives it the property of being Input Strictly Local (for k =3).

More generally, transformations describable with rules of the form A $\rightarrow$B/C___D that apply simultaneously can be modeled with Input Strictly Local FSTs provided that CAD is a finite set of strings (Chandlee, 2014; Chandlee & Heinz, forthcoming).

Output Strictly Local (OSL). OSL functions come in two varieties: left and right. Left and Right OSL functions are those in which the output string associated with each input symbol depends only on the input symbol itself and the last $k-1$ output symbols written. Whether an OSL function is left or right depends on whether it reads the input string from the left or from the right. (Left and Right ISL functions could be defined similarly, but those two classes would be equally expressive, which makes the distinction artificial.) The diagrams in Figure 12 illustrate how Left and Right OSL functions make computations. For Left (Right) OSL functions, the output of symbol x is u, and u only depends on x and the $k-1$ output symbols preceding (following) u (again $k=3$ in the example).

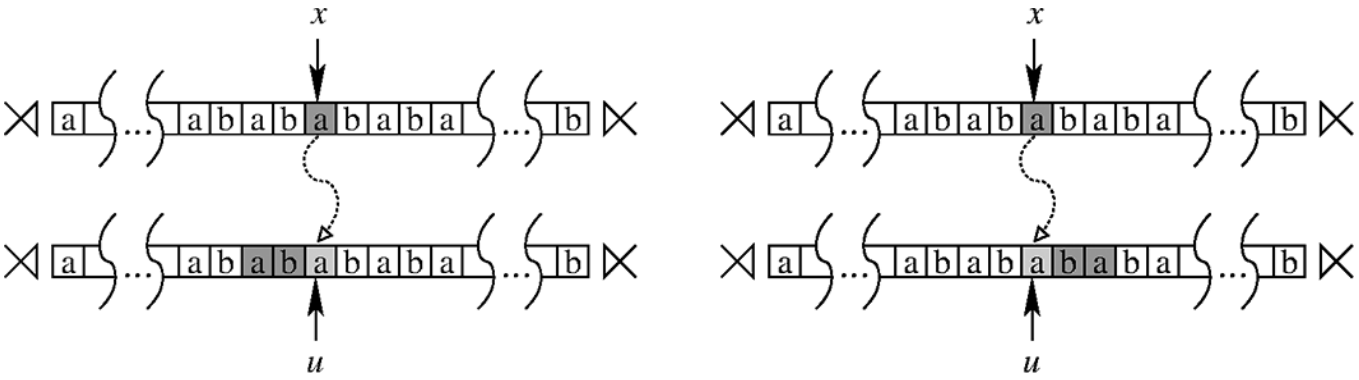


Figure 12. Left and Right Output Strictly Local functions.

As an example, consider an iterative nasal spreading process by which nasality spreads from a nasal segment to a following contiguous span of vowels and glides. So if the input form contains a sequence like NVVV (where N is a nasal and V is a vocalic segment), this sequence in the output will be $\mathrm{N\tilde{V}\tilde{V}\tilde{V}}$. Unlike the Input Strictly Local example of post-nasal obstruent voicing, because the vowels that trigger additional nasalization in $\mathrm{NVVV} \mapsto \mathrm{N\tilde{V}\tilde{V}\tilde{V}}$ are not nasalized in the input, the FST instead needs to keep track of the most recent *output*. This is what distinguishes Input Strictly Local and Output Strictly Local FSTs.

More generally, a k-Output Strictly Local FST is one in which the states correspond to sequences from Δ of length $k-1$, and the transitions are defined such that the FST is always in the state that matches the previous $k-1$ segments of the output (Chandlee, Eyraud, & Heinz, 2015). The example of nasal spreading is progressive, because the triggering nasalized segment precedes the target, and so it can be modeled with a FST that reads the string from left-to-right (i.e., a Left Output Strictly Local FST). In the case of regressive spreading, the analysis is the same except the string is read starting from the right (i.e., a Right Output Strictly Local FST). See Johnson (1972), Kaplan and Kay (1994), and Hulden (2009a) for more on modes of rule application and matching contexts in the input versus output.

Together the Input Strictly Local and Output Strictly Local FST classes can describe virtually all local phonological transformations, here characterized as those processes for which the target and triggering context form a contiguous substring of bounded length. As for long-distance/unbounded transformations, an Input Strictly Local or Output Strictly Local analysis

will not work in cases where a potentially unbounded number of segments can intervene between the target and trigger. The classic example is vowel harmony, in which one vowel harmonizes to another despite the presence of any number of intervening consonants. Though not describable with the very restrictive Input Strictly Local/Output Strictly Local FSTs, such long-distance processes still belong to well-defined subregular classes of regular relations.

Subsequential. Subsequential functions also come in two varieties: left and right. Left and Right subsequential functions are those in which the output string associated with each input symbol depends only on the input symbol itself and the state of the transducer as determined by the input read so far. Importantly, there are only finitely many states and the transducer can only be in one state at each moment, as Figure 13 illustrates. (Note that Sakarovitch [2009] and Filiot and Reynier [2016] use the term ‘sequential’ where we have used ‘subsequential’.)

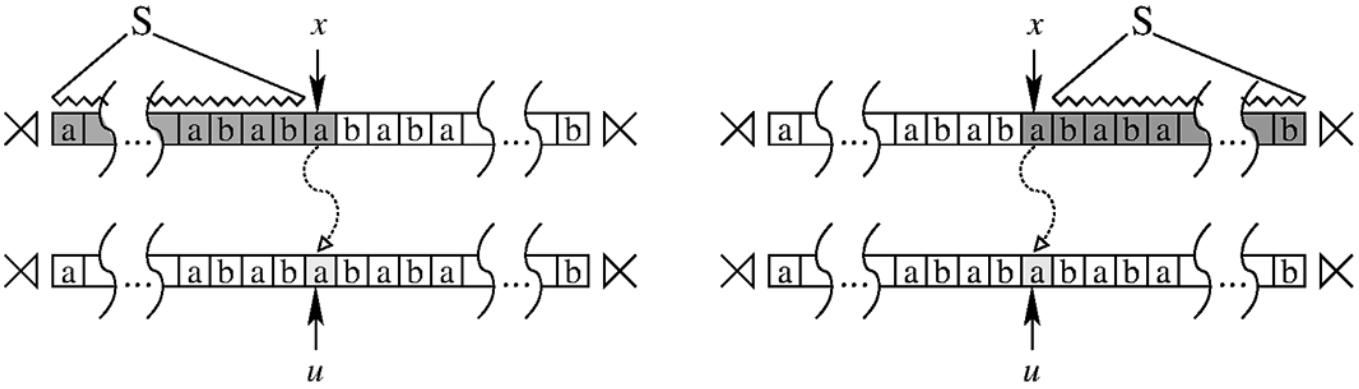


Figure 13. Left and Right Subsequential functions.

Gainor, Lai, and Heinz (2012) and Heinz and Lai (2013) establish that progressive vowel harmony can be modeled with a Left Subsequential transducer and regressive harmony can be described with a Right Subsequential transducer. Beyond vowel harmony, it has been shown that long-distance consonant agreement (Luo, 2013) and dissimilation (Payne, 2013) can also be described with Subsequential FSTs. As shown in Figure 9, the Subsequential FSTs properly include the Input Strictly Local and Output Strictly Local classes, but are themselves proper subclasses of the regular relations (Mohri, 1997).

However, it may be the case that the complexity of the Subsequential classes is actually more than what is needed for long-distance transformations. By conjecture, long-distance processes like vowel harmony and consonant agreement and dissimilation might belong to additional classes that are more restrictive than Left and Right Subsequential but less restrictive than Input Strictly Local and Output Strictly Local. Such classes are not represented in Figure 9 because they remain to be discovered. But additional subregular classes of functions can be established that correspond to other regions of the Subregular Hierarchy of languages in Figure 6. The Input Strictly Local and Output Strictly Local classes, for example, are based on the Strictly Local formal languages. As noted in §2.4, the Strictly Piecewise and Tier-based

Strictly Local formal language classes can represent long-distance phonotactic knowledge (Heinz, 2010a; Heinz, Rawal, & Tanner, 2011), so it's possible that the corresponding long-distance transformations will be shown to be Strictly Piecewise or Tier-based Strictly Local functions.

In sum, it has been shown that when it comes to the translation problem, the strong hypothesis is that phonological transformations are properly subregular. In particular, traditionally labeled 'local' transformations—here described as those transformations that involve contiguous substrings of bounded length—belong to the Input Strictly Local, Right Output Strictly Local, and Left Output Strictly Local classes of functions. The long-distance transformations that are excluded from the Input Strictly Local and Output Strictly Local classes can instead be represented with Left or Right Subsequential FSTs.

In addition to the as-yet undefined Strictly Piecewise and Tier-based Strictly Local function classes, another open question is to what extent transformations at the suprasegmental level support the subregular hypothesis. This discussion of transformations has focused on segmental phenomena, but there are also transformations of tone and stress patterns. Less is known about the computational complexity of suprasegmental transformations, but there is evidence that their computational nature is distinct from segmental phenomena (Jardine, 2016a).

4. The Learning Problem

There are several learning problems in phonology. Roughly, these problems are ones where the instance space are finite bodies of linguistic experience and answers are bodies of phonological knowledge. A learning algorithm solves this problem: it maps experience to grammars that represent phonological knowledge.

The problem of learning is commonly studied along one of two general approaches: the development of algorithms that provably solve well-defined problems and the development of computer programs that are run with some input data and whose resulting outputs are examined and measured with respect to some criteria. There are advantages and disadvantages to both approaches. Niyogi (2006) explains the differences this way in his book (which adopts the former method).

Another aspect of the book is its focus on mathematical models where the relationship between various objects may be formally (provably) studied. A complementary approach is to consider the larger class of computational models where one resorts to simulations. Mathematical models with their equations and proofs, and computational models with their equations and simulations provide different and important windows of insight into the phenomena at hand. In the first, one constructs idealized and simplified models but one can now reason precisely about the behavior of such models and therefore be very sure of one's conclusions. In the second, one constructs more realistic models but because of the complexity, one will need to resort to heuristic arguments and simulations. In summary, for

mathematical models the assumptions are more questionable but the conclusions are more reliable—for computational models, the assumptions are more believable but the conclusions more suspect.

Heinz and Riggle (2011, pp. 67–68) elaborate, explaining why computational learning theorists focus on the mathematical approach:

> having observed that an algorithm $\mathscr{A}$ and human subject $\mathscr{H}$ give similar responses for a particular set of test items $\mathscr{T}$ after being exposed to a set of training data $\mathscr{D}$, it is not clear what we can conclude about the relationship between $\mathscr{A}$ and $\mathscr{H}$ because they might wildly diverge for some other data $\mathscr{T}\prime$ and $\mathscr{D}\prime$. The goal of determining which properties of the data critically underlie learnability—or in this case the correlation between $\mathscr{A}$ and $\mathscr{H}$ is precisely why learning theory focuses mainly on the *properties of classes of languages* or the *general behavior of specific algorithms*, as opposed to the specific behavior of specific algorithms. (emphasis in original)

Here we briefly summarize both threads of research. Overviews of learning in phonological theory are given by Albright and Hayes (2011) and Heinz and Riggle (2011).

4.1. Formal Research

There are many ways the learning problem can be stated (Heinz & Riggle, 2011; Heinz, 2016). A detailed explanation of different learning paradigms is not provided here. Instead we assume that linguistic experience is composed solely of positive examples and that the algorithm must be able to make efficient computations.

There are three general problems. The *phonotactic learning problem* and the *transformation learning problem* can be viewed as steppingstones to the *grammar learning* problem, which is the ultimate goal. The phonotactic learning problem asks how a speaker learns what is and isn't a possible string in his or her language given a finite set of words that are in the language. The transformation learning problem asks how a speaker learns the map from underlying to surface representations given a finite set of (underlying form, surface form) pairs. The instance space of the grammar learning problem includes morpho-phonological paradigms of the kind we find in textbook phonology exercises (but including ones on much larger scales), and the answers are grammars that include a lexicon (the underlying forms of morphemes), a specification of how the morphemes combine to form words, and a specification of the phonological map that transforms underlying to surface forms.

There are formal results with respect to the first two problems, but none with respect to the third. Researchers have, however, developed computer programs that run simulations for all three problems.

For phonotactic knowledge, Heinz (2007, 2010b, 2010a) presents learning algorithms for both local and long-distance phonotactics. These algorithms crucially rely on the fact that such knowledge is properly subregular. Jardine and Heinz (2015b) also prove that constraints over phonological tiers can be learned even when the tier is not known a priori.

Chandlee, Heinz, and Eyraud (2014), Jardine, Chandlee, Eyraud, and Heinz (2014), and Chandlee, Eyraud, and Heinz (2015) present learners for transformations based on the defining properties of the Input Strictly Local and Output Strictly Local classes. The algorithm OSTIA (Onward Subsequential Transducer Inference Algorithm) (Oncina & Garcia, 1991) also provably learns the class of total, Left Subsequential functions from positive data, and, given the results reviewed in Section 3, can also be used for the transformation learning problem (see Gildea & Jurafsky, 1996).

There are also formal results for the transformation learning problem, when the output of the algorithm is an OT grammar. Tesar and Smolensky (1998) present Recursive Constraint Demotion, which provably and efficiently returns an OT grammar consistent with the data input to the algorithm (if such a grammar exists). Riggle (2009) shows that the set of OT grammars entailed by a set of constraints is also learnable under a different set of learning conditions (the Probably Approximately Correct learning paradigm). On the other hand, Magri (2013a) establishes the problem of finding an OT grammar consistent with the data that generates a smallest language and shows that it is intractable. Magri (2013b) also supplies a convergence proof for learning stochastic OT grammars (Boersma, 1997; Boersma & Hayes, 2001).

A Harmonic Grammar (HG) weights constraints instead of ranking them (Potts, Pater, Bhatt, & Becker, 2008). Pater (2008) shows how HG grammars will correctly converge to the correct one (if it exists); this approach is based on the perceptron learning algorithm (Rosenblatt, 1958). An important difference between HG and OT grammar learning is that the number of errors the learner of OT grammars will make is provably bounded whereas it is not for HG learners (Riggle, 2009; Heinz & Riggle, 2011).

With respect to the grammar learning problem, there are currently no general algorithmic results. One interesting approach relies on the hypothesis that phonological maps are structured in a way according to a similarity metric among representations. Let $d(w, w')$ be the relevant distance metric that measures the degree of disparity between w and w' according to a pre-determined similarity metric. A map is output-driven if $d(u', s) < d(u, s)$ and if underlying form u maps to surface form s then underlying form u' must also map to s (Tesar, 2014). Tesar shows that a learner based on this idea is useful in certain cases because it outputs accurate lexicons and grammars, but a general learnability theorem remains to be stated and proved.

4.2. Modeling Research

Many programs have been developed for the phonotactic learning problem. Many of these programs return representations of functions with co-domain $[0,1]$, which can be evaluated against human judgements collected from experiments (Coleman & Pierrehumbert, 1997; Hayes & Wilson, 2008; Albright, 2009; Daland et al., 2011). Gorman (2013) presents a program whose output is a representation of a function with co-domain $\{0,1\}$ that still accounts for human acceptability judgements (his analysis accounts for the gradient judgements with other factors).

Goldsmith and Riggle (2012) present experiments that suggest that vowel tiers can be induced from distributional information, which can also be used to learn constraints that operate over these tiers.

Dresher and Kaye (1990) present one of the first learning algorithms for learning stress patterns in words, which was later implemented by Gillis, Durieux, and Daelemans (1995). Other programs developed for learning stress patterns are explored in Goldsmith (1994), Gupta and Touretzky (1994), and Heinz (2009).

With respect to the learning of transformations, there have been some investigations of programs that learn phonological rules (Albright & Hayes, 2003; Calamaro & Jarosz, 2015). Within constraint-based formalisms there has been modeling work that builds on the ideas of Recursive Constraint Demotion. Boersma and Hayes (2001) explore a stochastic version of OT on some different language patterns. One wrinkle is the problem of learning when the phonological surface forms contain ‘hidden structure’. This means some aspects of the surface representation are not pronounced (such as metrical feet) and thus they are ’hidden’. A common approach to this problem modifies Recursive Constraint Demotion with a sub-procedure (Robust Interpretive Parsing), which uses the current grammatical hypothesis to guess the hidden structure and then continue learning (Tesar & Smolensky, 2000; Jarosz, 2013).

With respect to the grammar learning problem, one approach has been based on maximum likelihood estimation (in particular it adapts the expectation-maximization method) to simultaneously learn both the grammar and the lexicon (Jarosz, 2006b, 2006a). Another approach uses loopy belief propagation in Bayesian networks to identify underlying forms and the phonological transformation, which is modeled with a finite-state transducer (Cotterell, Peng, & Eisner, 2015). Further research along these lines should help us to better understand the limits of these techniques. Both methods represent progress and are successful on certain cases; future work may aim to characterize those cases where these methods are guaranteed to work.

5. Other Related Topics

What if the representations of phonological forms changes from strings to something else? What does the computational analysis of phonological patterns presented in Sections 2 and 3 mean for phonological theories? Do they make any psycholinguistic predictions? And finally, how do the results of computational phonology inform real-world applications?

5.1. Representations and Data Structures

The previous studies represent words with strings, but theoretical phonology often invokes richer representations. One of the most celebrated examples is autosegmental phonology (Leben, 1973; Goldsmith, 1979). These representations can be formalized as graphs (Coleman & Local, 1991).

Figure 14 shows an autosegmental representation of some words from Mende, a tonal language. Some words like ‘rice’ [mbă] have a rising tone, as indicated by the diacritic on the vowel. Other vowels in other words only carry low or high tones. For example the word ‘sling’ [ndàvúlá] has a low tone on the first vowel and high tones on the other vowels as indicated by the diacritics. Figure 14 shows non-string representations of these words. The defining feature of these graphs is that the tones are *autonomous* from the segmental level of the features. (Our notation comes from Jardine & Heinz, 2015a; Jardine, 2016a.)

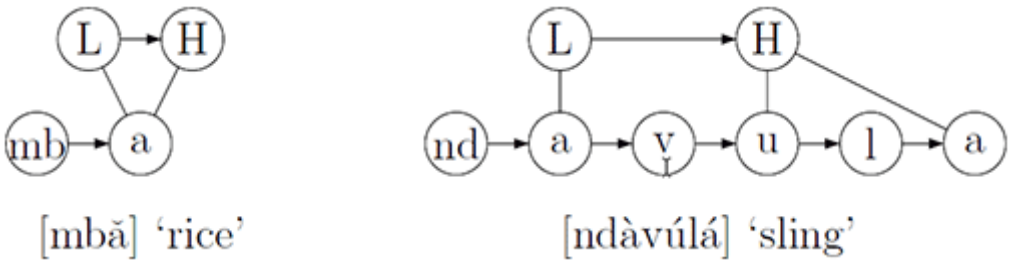


Figure 14. Autosegmental representations in Mende.

Autosegmental representations similar to the ones in Figure 14 have been employed to account for other aspects of phonological theory, including long-distance and local assimilation processes (Clements, 1976; Poser, 1982).

An important question is how are constraints over such representations to be defined? How are phonological transformations over them to be defined? There has been substantial work on this using multi-tape finite-state transducers (Kornai, 1995). Another approach being explored is based on MSO-definable transductions (Engelfriet & Hoogeboom, 2001). Jardine (2016b) argues that Strictly-Local-like constraints over autosegmental representations provides the best phonological theory of tone in terms of characterizing the attested typology and in obtaining purchase on the learning problem.

5.2. Implications for Phonological Theories

Some implications of the computational analysis presented in Sections 2 and 3 for phonological theories like the ones presented in Chomsky and Halle (1968) and Prince and Smolensky (2004) are discussed.

As mentioned in Section 3, grammars that apply rewrite rules in a particular order to underlying forms are able to express phonological transformations for all known languages. So these theories do not undergenerate the known typology. On the other hand, because these grammars are equivalent in expressivity to the regular relations, there is a sense in which they overgenerate. For example, a logically possible regular relation is one that changes underlying sibilants so that they agree on the surface only if there are an even number of underlying sibilants but not if there are an odd number.

On the other hand, consider a theory that posits that phonological transformations must be Strictly Local (ISL, LOSL, or ROSL), which we will call Subregular Theory. Subregular Theory will not overgenerate this bizarre even/odd sibilant harmony transformation. However, as mentioned, no Strictly Local function can compute the attested sibilant harmony transformations. Thus Subregular Theory is currently incomplete. It needs a functional equivalent to the Tier-based Strictly Local or the Strictly Piecewise classes of stringsets.

Koskenniemi (1983) proposes a theory of morpho-phonological transformations based on constraints (see also Yli-Jyrä & Koskenniemi, 2006). This theory posits constraints on *relations* and an underlying form maps to a surface form if it violates no constraint. In this sense, the constraints are inviolable and language-specific. This theory has been used to develop large-scale implementations of the morpho-phonology for several languages (Arppe, 2005). It turns out that two-level morpho-phonology is also equally expressive to regular relations. Thus, they overgenerate to the same extent as rule-based grammars.

Next we turn to Optimization-based theories such as Optimality Theory, Harmonic Grammar (Potts, Pater, Bhatt, & Becker, 2008), Harmonic Serialism (McCarthy, 2008), and Maximum Entropy (Goldwater & Johnson, 2003; Hayes & Wilson, 2008). A well-known problem for these theories is phonological opacity (Idsardi, 1998; Kager, 1999; Idsardi, 2000; Baković, 2007; McCarthy, 2007). In this way, optimization theories undergenerate.

The issue has less to do with optimization itself than it does with the division of constraints into the two varieties known as markedness and faithfulness constraints (Buccola, 2013). Several strategies reviewed by McCarthy (2007) introduce new constraint types to address this problem. For example, if two-level constraints are adopted then this undergeneration problem goes away (see Kager, 1999, chap. 9, for additional discussion). Of course, once two-level constraints are adopted, then the constraints can be inviolable (Koskenniemi, 1983), obviating the need for optimization (though they then must be language-specific).

Optimization theories can also generate non-regular patterns (Frank & Satta, 1998; Riggle, 2004; Gerdemann & Hulden, 2012) with very simple constraints. Majority Rules harmony is one example (Baković, 2000; Finley, 2008; Heinz & Lai, 2013). This is not controversial. While

it may be possible to eliminate some overgeneration with carefully chosen constraints, it is not at all clear how to ensure this. This overgeneration seems to be a direct consequence of optimization itself. The ability to examine the entire phonological representation and choose the best output representation involves a kind of *global* comparison and computation. This strikes us as very different from the Strictly Local computations that operate on local sub-structures of the representations independently of the other sub-structures.

Each of the above theories overgenerates in some manner. How does one compare theories that overgenerate in different ways? One answer is to choose the one that overgenerates in a computationally simple way. In this respect, a theory that is sufficiently expressive for phonology but fails to generate every regular pattern (the goal of Subregular Theory) is better than one that is sufficiently expressive but generates every regular pattern (rule-based theories), which in turn is better than one that generates non-regular patterns (optimization-based theories).

5.3. Psycholinguistic Predictions

Rogers and Pullum (2011) and Jäger and Rogers (2012) discuss psychological predictions that the computational complexity of natural language patterns make. In principle these can be examined with artificial grammar learning experiments where subjects are trained on some forms that are generated according to some grammar of some complexity. Subjects are then tested to see if they accept or reject novel forms according to the underlying generative grammar. As Rogers and Pullum (2011) and Jäger and Rogers (2012) point out, such experiments must be designed carefully and their results must also be interpreted with care. Moreton and Pater (2012a, 2012b) discuss similar ideas, but not in the context of formal language theory.

Rogers et al. (2013) provide cognitive correlates for the classes in the Subregular Hierarchy (Figure 6). Each subregular class corresponds to a particular kind of memory model. It follows that lower classes in the hierarchy have simpler memory models than higher levels in the hierarchy. It is reasonable to expect then that patterns that can only be described at the higher levels ought to be more difficult to learn than those that can be described at the lower levels.

The first artificial grammar learning experiments that bear on computational complexity were conducted by Finley (2008) and Finley and Badecker (2009), who compared a non-regular pattern (Majority Rules vowel harmony) with a regular one (left-to-right directional harmony). They found that subjects had no difficulty learning the vowel harmony pattern that is regular but they could not learn the Majority Rules pattern with a similar training regimen.

In a series of artificial grammar learning experiments, Lai (2015) compared the learning of a Locally Testable pattern with a Strictly Piecewise pattern and found that subjects could more easily learn the Strictly Piecewise pattern than the Locally Testable pattern. Recently Hwangbo (2015) conducted artificial grammar learning experiments to compare the

learnability of a Locally Testable pattern with a Non-counting pattern. She too found that subjects had more difficulty learning the pattern higher in the Subregular Hierarchy (the Non-counting pattern) than the one lower (the Locally Testable one).

The aforementioned results are the only ones to our knowledge that have examined the predictions of the Subregular Hierarchy, and they are consistent with those predictions.

5.4 Applications

The well-supported hypotheses that phonotactic knowledge can be represented by regular sets and phonological transformations by regular relations have had a significant impact on the development of software for automatic language analysis.

Two industrial-scale toolkits for the development of morpho-phonological language models are in use by both theoretical and computational linguists. The first, xfst, was produced by Xerox corporation (Beesley & Karttunen, 2003) and the second, foma, is an open-source alternative developed by Hulden (2009a, 2009b).

These toolkits extend the regular expression syntax and semantics for sets of strings to relations over strings. This allows researchers to describe regular relations in a high-level language, which the software then compiles into low-level finite-state transducers. Readers are encouraged to consult Beesley and Karttunen (2003) for details.

The xfst and foma software packages allow researchers to implement phonological analyses in various theoretical traditions, including rule-based theories (Chomsky & Halle, 1968; Gerdemann & Hulden 2015), Optimality Theory (under certain conditions) (Prince & Smolensky, 2004; Karttunen, 1998), and two-level morpho-phonology (Koskenniemi, 1983). Such toolkits allow analysts to be more confident that their analyses are without error (Karttunen, 2006).

Riggle (2004) provides another finite-state approach to computing maps defined by OT grammars. Finite-state models also form the basis of the constraint-based maximum entropy phonotactic learner presented in Hayes and Wilson (2008).

More broadly, finite-state machines typically form the backbone of models for speech and language processing (Mohri, 1997; Carson-Berndsen, 1998; Mohri, 2005; Roark & Sproat, 2007). This is because the membership and translation problems have efficient solutions when the problems are Regular. It is also because several operations over relations (such as union and composition) can be computed when the relations are Regular. This “finite-state calculus” makes it possible that linguistic generalizations that are Regular may be factored into simpler, distinct parts that work together to model language.

6. The Computational Nature of Phonology

The nature of the computations entailed by phonological knowledge has revealed that there is strong support for the hypothesis that “they are regular but not finite.” These insights have played a role in the development of technologies that many of us now carry on our phones. As better *subregular* computational characterizations of phonological knowledge are obtained we can expect better theories of phonology whose predictions more tightly match the known typology, more psycholinguistic predictions that can be studied in the lab, as well as better algorithmic solutions to learning problems in phonology.

References

Albright, A. (2009). Feature-based generalisation as a source of gradient acceptability. *Phonology*, *26*(1), 9–41.

Albright, A., & Hayes, B. (2003). Rules vs. analogy in English past tenses: A computational/experimental study. *Cognition*, *90*, 119–161.

Albright, A., & Hayes, B. (2011). Learning and learnability in phonology. In J. Goldsmith, J. Riggle, & A. Yu (Eds.), *Handbook of phonological theory* (pp. 661–690). Chichester, U.K.: Wiley-Blackwell.

Arppe, A. (2005). Kimmo Koskenniemi’s first 60 years. In A. Arppe, L. Carlson, K. Lindn, J. Piitulainen, M. Suominen, M. Vainio, et al. (Eds.), *The very long way from basic linguistic research to commercially successful language business: The case of two-level morphology* (pp. 2–17). Stanford, CA: CSLI Publications.

Baković, E. (2000). *Harmony, dominance and control* (Ph.D. thesis). Rutgers University.

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

Beesley, K., & Karttunen, L. (2003). *Finite state morphology*. Stanford, CA: CSLI Publications.

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

Boersma, P., & Hayes, B. (2001). Empirical tests of the gradual learning algorithm. *Linguistic Inquiry*, *32*, 45–86.

Bromberger, S., & Halle, M. (1989). Why phonology is different. *Linguistic Inquiry*, *20*, 51–70.

Buccola, B. (2013). On the expressivity of optimality theory versus ordered rewrite rules. In G. Morrill & M. Nederhof (Eds.), *Proceedings of formal grammar 2012 and 2013*, Vol. 8306 of *Lecture notes in computer science 8036* (pp. 142–158). Berlin: Springer-Verlag.

Büchi, J. R. (1960). Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, *6*(1–6), 66–92.

Calamaro, S., & Jarosz, G. (2015, April). Learning general phonological rules from distributional information: A computational model. *Cognitive Science*, *39*(3), 647–666.

Carson-Berndsen, J. (1998). *Time map phonology: Finite state models and event logics in speech recognition*. Dordrecht, The Netherlands: Kluwer Academic.

Chandlee, J. (2014). *Strictly local phonological processes* (Ph.D. thesis). University of Delaware.

Chandlee, J., Eyraud, R., & Heinz, J. (2015, July). Output strictly local functions. In M. Kuhlmann, M. Kanazawa, & G. M. Kobele (Eds.), *Proceedings of the 14th Meeting on the Mathematics of Language (MOL 2015)* (pp. 112–125). Association for Computational Linguistics, Chicago.

Chandlee, J., & Heinz, J. (forthcoming). Strict locality and phonological maps. *Linguistic Inquiry*.

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

Chomsky, N., & Halle, M. (1968). *The sound pattern of English*. New York: Harper & Row.

Clements, G. (1976). Vowel harmony in nonlinear generative phonology: An autosegmental model. Bloomington: Indiana University Linguistics Club.

Coleman, J., & Local, J. (1991). The “no crossing constraint” in autosegmental phonology. *Linguistics and Philosophy*, *14*(3), 295–338.

Coleman, J. S., & Pierrehumbert, J. (1997). Stochastic phonological grammars and acceptability. In *Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology* (pp. 49–56). Somerset, NJ: Association for Computational Linguistics.

Cook, E. D. (1978). Palatalizations and related rules in Sarcee. In E. D. Cook & J. Kaye (Eds.), *Linguistic studies of Native Canada* (pp. 19–35). Vancouver, BC: University of British Columbia Press.

Cook, E. D. (1984). *A Sarcee grammar*. Vancouver, BC: University of British Columbia Press.

Cotterell, R., Peng, N., & Eisner, J. (2015). Modeling word forms using latent underlying morphs and phonology. *Transactions of the Association for Computational Linguistics*, *3*, 433–447.

Daland, R. (2014). What is computational phonology? *Loquens*, *1*(1), e004.

Daland, R. (2015). Long words in maximum entropy phonotactic grammars. *Phonology*, *32*(3), 353–383.

Daland, R., Hayes, B.White, J.Garellek, M.Davis, A., & Normann, I. (2011). Explaining sonority projection effects. *Phonology*, *28*(197), 197–234.

de Lacy, P. (2011). Markedness and faithfulness constraints. In M. V. Oostendorp, C. J. Ewen, E. Hume, & K. Rice (Eds.), *The Blackwell companion to phonology*. Malden, MA: Wiley-Blackwell.

Dresher, E., & Kaye, J. (1990). A computational learning model for metrical phonology. *Cognition*, *34*, 137–195.

Droste, M., & Gastin, P. (2009). Weighted automata and weighted logics. In M. Droste, W. Kuich, & H. Vogler (Eds.), *Handbook of Weighted Automata* (pp. 175–212). Monographs in Theoretical Computer Science. Berlin: Springer.

Droste, M., Kuich, W., & Vogler, H. (Eds.). (2009). *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. Berlin: Springer.

Eisner, J. (2003). Simpler and more general minimization for weighted finite-state automata. In M. Hearst & M. Ostendorf (Eds.), *Proceedings of the Joint Meeting of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics* (HLT-NAACL 2003) (pp. 64–71). Association for Computational Linguistics, Edmonton, Canada.

Enderton, H. B. (2001). *A mathematical introduction to logic* (2d ed.). New York: Academic Press.

Engelfriet, J., & Hoogeboom, H. J. (2001, April). MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Logic*, *2*(2), 216–254.

Filiot, E., & Reynier, P. A. (2016, July). Transducers, logic and algebra for functions of finite words. *SIGLOG News*, *3*(3), 4–19.

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

Finley, S., & Badecker, W. (2009). Right-to-left biases for vowel harmony: Evidence from artificial grammar. In A. Shardl, M. Walkow, & M. Abdurrahman (Eds.), *Proceedings of the 38th North East Linguistic Society Annual Meeting, October 26–28, 2007, Ottawa, ON*. Vol. 1 (pp. 269–282). Amherst, MA: GLSA.

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

Gainor, B., Lai, R., & Heinz, J. (2012). Computational characterizations of vowel harmony patterns and pathologies. In J. Choi, E. A. Hogue, J. Punske, D. Tat, J. Schertz, & A. Trueman (Eds.), *WCCFL 29: Proceedings of the 29th West Coast Conference on Formal Linguistics*, April 22–24, 2011, University of Arizona (pp. 63–71). Somerville, MA: Cascadilla.

Gerdemann, D., & Hulden, M. (2012, July). Practical finite state optimality theory. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing, July 23–25, University of Basque, Donostia-San Sebastián* (pp. 10–19). Stroudsburg, PA: Association for Computational Linguistics.

Gerdemann, D., & Hulden, M. (2015). Grammar design with multi-tape automata and composition. In *Proceedings of the 12th International Workshop on Finite State Methods and*

Natural Language Processing, June 22–24, Dusseldorf, Germany. Stroudsburg, PA: Association for Computational Linguistics.

Gildea, D., & Jurafsky, D. (1996). Learning bias and phonological-rule induction. *Computational Linguistics*, *24*(4), 497–530.

Gillis, S., Durieux, G., & Daelemans, W. (1995). A computational model of P&P: Dresher & Kaye (1990) revisited. In F. Wijnen & M. Verrips (Eds.), *Approaches to parameter setting* (pp. 135–173). Amsterdam: Instituut voor Algemene Taalwetenschap.

Goldsmith, J. (1979). *Autosegmental phonology*. New York: Garland.

Goldsmith, J. (1994). A dynamic computational theory of accent systems. In J. Cole & C. Kisseberth (Eds.), *Perspectives in Phonology* (pp. 1–28). Stanford, CA: Center for the Study of Language and Information.

Goldsmith, J., & Riggle, J. (2012). Information theoretic approaches to phonological structure: the case of Finnish vowel harmony. *Natural Language and Linguistic Theory*, *30*(3), 859–896.

Goldwater, S., & Johnson, M. (2003). Learning OT constraint rankings using a maximum entropy model. In J. Spenader, A. Eriksson, & O. Dahl (Eds.), *Proceedings of the Stockholm Workshop on Variation within Optimality Theory* (pp. 111–120). April 26–27, Department of Linguistics, Stockholm University, Sweden.

Goodman, J. (1999, December). Semiring parsing. *Computational Linguistics*, *25*(4), 573–606.

Gorman, K. (2013). *Generative phonotactics*. University of Pennsylvania dissertation.

Graf, T. (2010a). Comparing incomparable frameworks: A model theoretic approach to phonology. *University of Pennsylvania Working Papers in Linguistics*, *16*(2), Article 10.

Graf, T. (2010b). *Logics of phonological reasoning* (Master's thesis). University of California, Los Angeles.

Gupta, P., & Touretzky, D. (1994). Connectionist models and linguistic theory: Investigations of stress systems in language. *Cognitive Science*, *18*(1), 1–50.

Halle, M., & Vergnaud, J. R. (1987). *An essay on stress*. Cambridge, MA: The MIT Press.

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

Hayes, B., & Wilson, C. (2008). A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, *39*, 379–440.

Heinz, J. (2007). *The inductive learning of phonotactic patterns* (Doctoral thesis). University of California, Los Angeles.

Heinz, J. (2009). On the role of locality in learning stress patterns. *Phonology*, *26*(2), 303–351.

Heinz, J. (2010a). Learning long-distance phonotactics. *Linguistic Inquiry*, *41*(4), 623–661.

Heinz, J. (2010b). String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11–16, Uppsala, Sweden* (pp. 897–906). Stroudsburg, PA: Association for Computational Linguistics.

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

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

Heinz, J. (2016). Computational theories of learning and developmental psycholinguistics. In J. Lidz, W. Synder, & J. Pater (Eds.), *The Oxford handbook of developmental linguistics*. Oxford: Oxford University Press.

Heinz, J., & Lai, R. (2013, August). Vowel harmony and subsequentiality *<http://www.aclweb.org/anthology/W13-3006>*. In A. Kornai & M. Kuhlmann (Eds.), *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)* (pp. 52–63), Association for Computational Linguistics, Sofia, Bulgaria.

Heinz, J., Rawal, C., & Tanner, H, G. (2011). Tier-based Strictly Local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, June 19–24, Portland, OR* (pp. 58–64). Red Hook, NY: Curran Associates.

Heinz, J., & Riggle, J. (2011). Learnability. In M. van Oostendorp, C. Ewen, B. Hume, & K. Rice (Eds.), *Blackwell companion to phonology*. Malden, MA: Wiley-Blackwell.

Hopcroft, J., Motwani, R., & Ullman, J. (2001). *Introduction to Automata Theory, languages, and computation*. Boston: Addison-Wesley.

Hulden, M. (2009a). *Finite-state machine construction methods and algorithms for phonology and morphology* (Ph.D. thesis). University of Arizona.

Hulden, M. (2009b). Foma: A finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, March 30–April 3, Athens, Greece* (pp. 29–32). Morristown, NJ: Association for Computational Linguistics.

Hwangbo, H. J. (2015). *Learnability of two vowel harmony patterns with neutral vowels*. Talk presented at the 3rd Annual Meeting of Phonology, Vancouver, BC.

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

Idsardi, W. J. (2000). Clarifying opacity. *Linguistic Review*, *17*, 337–350.

Jäger, G., & Rogers, J. (2012). Formal language theory: Refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B*, *367*(1598), 1956–1970.

Jardine, A. (2016a). Computationally, tone is different. *Phonology*, *33*(2), 247–283.

Jardine, A. (2016b). *Locality and non-linear representations in tonal phonology* (Doctoral thesis). University of Delaware.

Jardine, A., Chandlee, J.Eyraud, R., & Heinz, J. (2014, September). Very efficient learning of structured classes of subsequential functions from positive data. In A. Clark, M. Kanazawa, & R. Yoshinaka (Eds.), *Proceedings of the Twelfth International Conference on Grammatical Inference (ICGI 2014), September 17–19, Kyoto, Japan*, Vol. 34 (pp. 94–108). JMLR: Workshop and Conference Proceedings <*http://www.jmlr.org/proceedings/*>.

Jardine, A., & Heinz, J. (2015a, July). A concatenation operation to derive autosegmental graphs. In *Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015), July 25–26, Chicago* (pp. 139–151). Stroudsburg, PA: Association for Computational Linguistics.

Jardine, A., & Heinz, J. (2015b, April). Learning Tier-based Strictly 2-Local Languages. *Transactions of the Association for Computational Linguistics*, *4*(2307–387X), 87–98. Accepted with minor revisions.

Jarosz, G. (2006a). *Rich lexicons and restrictive grammars—maximum likelihood learning in Optimality Theory* (Doctoral thesis). Johns Hopkins University.

Jarosz, G. (2006b). Richness of the base and probabilistic unsupervised learning in optimality theory. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology at HLT-NAACL, June 8, New York City* (pp. 50–59). Stroudsburg, PA: Association for Computational Linguistics.

Jarosz, G. (2013). Learning with hidden structure in optimality theory and harmonic grammar: Beyond robust interpretive parsing. *Phonology*, *30*(1), 27–71.

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

Jurafsky, D., & Martin, J. (2008). *Speech and language processing: An introduction to natural language processing, speech recognition, and computational linguistics* (2d ed.). Upper Saddle River, NJ: Prentice-Hall.

Kager, R. (1999). *Optimality theory*. Cambridge, U.K.: Cambridge University Press.

Kaplan, R., & Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, *20*(3), 331–378.

Karttunen, L. (1993). Finite-state constraints. In J. Goldsmith (Ed.), *The last phonological rule* (chap. 6). Chicago: University of Chicago Press.

Karttunen, L. (1998). The proper treatment of optimality in computational phonology. In L. Karttunen & K. Oflazer (Eds.), *FSMNLP'98, International Workshop on Finite-State Methods in Natural Language Processing, June 30–July 1: Bilkent University, Ankara, Turkey* (pp. 1–12). Stroudsburg, PA: Association for Computational Linguistics.

Karttunen, L. (2006). *The insufficiency of paper-and-pencil linguistics: The case of Finnish prosody*. Rutgers Optimality Archive #818-0406.

Kaye, J., Lowenstamm, J., & Vergnaud, J. R. (1990). Constituent structure and government in phonology. *Phonology*, *7*, 193–231.

Kenstowicz, M., & Kisseberth, C. (1979). *Generative phonology*. New York: Academic Press.

Kleene, S. (1956). Representation of events in nerve nets. In C. Shannon & J. McCarthy (Eds.), *Automata Studies* (pp. 3–40). Princeton, NJ: Princeton University Press.

Kornai, A. (1995). *Formal phonology*. Outstanding Dissertations in Linguistics. New York: Garland Publishing.

Koskenniemi, K. (1983). *Two-level morphology*. Publication 11, Department of General Linguistics. Helsinki: University of Helsinki.

Lai, R. (2015). Learnable vs. unlearnable harmony patterns. *Linguistic Inquiry*, *46*(3), 425–451.

Leben, W. (1973). *Suprasegmental phonology* (Doctoral thesis). Massachusetts Institute of Technology.

Luo, H. (2013). *Long-distance consonant harmony and subsequentiality* (Unpublished manuscript). University of Delaware.

Magri, G. (2013a). The complexity of learning in OT and its implications for the acquisition of phonotactics. *Linguistic Inquiry*, *44*(3), 433–468.

Magri, G. (2013b). HG has no computational advantages over OT: Toward a new toolkit for computational OT. *Linguistic Inquiry44*(4), 569–609.

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

McCarthy, J. J. (2008). The gradual path to cluster simplification. *Phonology*, *25*(2), 271–319.

McNaughton, R., & Papert, S. (1971). *Counter-Free Automata*. Cambridge, MA: MIT Press.

Mitchell, T. F. (1960). Prominence and syllabification in arabic. *Bulletin of the School of Oriental and African Studies*, *23*, 369–389.

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

Mohri, M. (2005). Statistical natural language processing. In M. Lothaire (Ed.), *Applied Combinatorics on Words*. Cambridge, U.K.: Cambridge University Press.

Moreton, E., & Pater, J. (2012a). Structure and substance in artificial-phonology learning: Part I, structure. *Language and Linguistics Compass*, *6*(11), 686–701.

Moreton, E., & Pater, J. (2012b). Structure and substance in artificial-phonology learning: Part II, substance. *Language and Linguistics Compass*, *6*(11), 702–719.

Niyogi, P. (2006). *The computational nature of language learning and evolution*. Cambridge, MA: MIT Press.

Odden, D. (2014). *Introducing phonology* (2d ed.). Cambridge, U.K.: Cambridge University Press.

Oncina, J., & Garcia, P. (1991). *Inductive learning of subsequential functions*. Technical Report DSIC II-34, University Politécnia de Valencia.

Pater, J. (2004). Austronesian nasal substitution and other NC effects. In J. McCarthy (Ed.), *Optimality theory in phonology: A reader* (pp. 271–289). Oxford: Blackwell.

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

Payne, A. (2013). Dissimilation as a subsequential process. In J. Iyer & L. Kusmer (Eds.), *NELS 44: Proceedings of the 44th Meeting of the North East Linguistic Society*, 2 (pp. 79–90). Amherst, MA, GLSA (Graduate Linguistics Student Association).

Poser, W. (1982). Phonological representation and action-at-a-distance. In H. van der Hulst & N. Smith (Eds.), *The structure of phonological representations* (pp. 121–158). Dordrecht, The Netherlands: Foris.

Potts, C., Pater, J., Bhatt, R., & Becker, M. (2008). *Harmonic grammar with linear programming: From linear systems to linguistic typology*. Rutgers Optimality Archive ROA-984.

Prince, A., & Smolensky, P. (2004). Optimality theory: Constraint interaction in generative grammar. Malden, MA: Blackwell Publishing.

Riggle, J. (2004). *Generation, recognition, and learning in Finite State Optimality Theory* (Doctoral thesis). University of California, Los Angeles.

Riggle, J. (2009). The complexity of ranking hypotheses in optimality theory. *Computational Linguistics*, *35*(1), 47–59.

Roark, B., & Sproat, R. (2007). *Computational approaches to morphology and syntax*. Oxford: Oxford University Press.

Rogers, J., Heinz, J., Fero, M., Hurst, J., Lambert, D., & Wibel, S. (2013). Cognitive and sub-regular complexity. In G. Morrill & M. J. Nederhof (Eds.), *Formal grammar* (pp. 90–108). Lecture Notes in Computer Science 8036. Heidelberg, Germany: Springer.

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

Rose, S., & Walker, R. (2004). A typology of consonant agreement as correspondence. *Language*, *80*, 475–531.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–408.

Sakarovitch, J. (2009). *Elements of Automata Theory*. Cambridge, U.K.: Cambridge University Press. Translated by Reuben Thomas from the 2003 edition published by Vuibert, Paris.

Sapir, E., & Hojier, H. (1967). *The phonology and morphology of the Navaho language*. University of California Publications 50, Berkeley: University of California.

Savitch, W. (1982). *Abstract machines and grammars*. Boston: Little, Brown.

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

Scott, D., & Rabin, M. (1959). Finite automata and their decision problems. *IBM Journal of Research and Development*, *5*(2), 114–125.

Sipser, M. (1997). *Introduction to the Theory of Computation*. Boston: PWS Publishing Company.

Tesar, B. (2014). *Output-driven phonology*. Cambridge, U.K.: Cambridge University Press.

Tesar, B., & Smolensky, P. (1998). Learnability in optimality theory. *Linguistic Inquiry*, *29*, 229–268.

Tesar, B., & Smolensky, P. (2000). *Learnability in Optimality Theory*. Cambridge, MA: MIT Press.

van der Hulst, H. (2011). Dependency-based phonologies. In J. A. Goldsmith, J. Riggle, & A. C. L. Yu (Eds.), *The Blackwell handbook of phonological theory* (pp. 533–570). Chichester, U.K.: Wiley-Blackwell.

Viljoen, J. J. (1973). *Manual for Ndonga: Part 1*. Pretoria, South Africa: University of South Africa.

Yli-Jyrä, A., & Koskenniemi, K. (2006). Compiling generalized two-level rules and grammars. In *FinTAL* (pp. 174–185). Lecture Notes in Artificial Intelligence 4139. Berlin: Springer-Verlag.

Related Articles

Variation in Phonology

Computational Approaches to Morphology

Morphology and Phonotactics