Complexity of the Simplified Segmental Phonology

Eric Sven Ristad[1]

Research Report CS-TR-388-92
August 1992

Abstract

This article presents detailed complexity analysis of the encoding and decoding problems for the segmental phonology. The segmental model is a generative theory of phonological dependencies. As such, it poses two computational problems. The first computational problem is the problem of actually encoding the dependencies of a given phonology into a given representation of phonological knowledge (the encoding problem). The second computational problem is the problem of deciding whether a given phonological representation encodes the dependencies of a given phonology (the decoding problem).

We begin by proving that the encoding and decoding problems are both undecidable in the segmental model. Next, we motivate a simplified segmental model, that more accurately models the phonological dependencies actually proposed in the phonological literature. The simplified segmental model is a more restricted and more natural reformalization of the segmental model. To conclude, we prove that the encoding and decoding problems are NP-complete in this simplified segmental model.

[1]This paper was revised in May 1993 and submitted for journal publication. The author is particularly grateful to Morris Halle, Michael Kenstowicz, and an anonymous CL reviewer for their assistance and constructive criticisms. The author has been supported by a Young Investigator Award IRI-9258517 from the National Science Foundation.

Linguistic sounds contain predictable information. For example, English speakers invariably aspirate a voiceless stop in the onset of syllable, but only when the onset is nonbranching. This is illustrated by minimal pairs such as [k^{h}ab]~[gab] (voiceless [k] versus voiced [g]), [p^{h}at]~[bat] (voiceless [p] versus voiced [b]), [p^{h}it]~[spit] (voiceless [p] in nonbranching onset [p] versus branching onset [sp]), and [k^{h}it]~[skit] (voiceless [k] in nonbranching onset [k] versus branching onset [sk]). The raised 'h' indicates aspiration, that is, that the segment is pronounced with a slight exhalation of air. Moreover, English vowels are lengthened before voiced consonants such as [b] or [g] (contrast the articulations of *cab*~*cap*, *bag*~*back*, and so forth). A sequence of primitive speech sounds is possible in given language only if it encodes the phonological dependencies of that language. Because such phonological dependencies are part of the language user's knowledge of his language, they must be represented in the language user and by every empirically adequate theory of language.

Generative phonology is an attempt to formally model the language user's knowledge of phonological dependencies. As such, every theory of generative phonology poses two computational problems. The first computational problem is the problem of encoding the dependencies of a given phonology into a given phonological representation (the encoding problem). The second computational problem is the problem of determining whether or not the dependencies of a given phonology are encoded in a given phonological representation (the decoding problem).[2] The central goal of this article is to improve our understanding of the structure and complexity of phonological dependencies. To this end, we present a detailed complexity analysis of the encoding and decoding problems for the segmental model of phonology [4], the most influential phonological theory of this century.

In the segmental model of phonology, phonological knowledge is represented as a string of segments. An underlying form is a segmental string that does not encode the dependencies of the phonology; it represents the true, unpredictable information content of a given surface form.[3] Underlying forms are constructed by combining (typically with concatenation or substitution) the underlying phonological forms of an utterance's constituent morphemes. Phonological dependencies are represented as rewriting rules. Dependencies are encoded in a given segmental string by applying a sequence of rewriting rules to that segmental string. That is, the grammar G of rewriting rules derives a

[2] The encoding and decoding problems are more widely known as the generation and recognition problems, respectively. The standard terminology incorrectly suggests that a generative theory is a model of the mental computations performed by the language user in the production and comprehension of utterances. Rather, a generative theory is a model of how the language user's knowledge of his language is mentally represented. We have therefore revised the standard terminology in order to clarify the distinction between (i) computations attributed to language users and (ii) computations performed by a generative theory in order to enumerate mental representations attributed to language users.

[3] This is true for all prominent generative theories of phonology, not just modern "underspecification" theories, such as [1].

surface form $[s]$ from its corresponding underlying form $/u/$ by repeated application of rewriting rules to the intermediate forms $i_1, i_2, \ldots$ of the derivation, as shown in (1):

$$G : /u/ \rightarrow /i_1/ \rightarrow /i_2/ \rightarrow \ldots \rightarrow [s] \tag{1}$$

Each rewriting rule represents a natural class of predictable phonological information. Continuing with our initial example, the phonological grammar of an English speaker must contain rules for aspiration and lengthening. The aspiration rule rewrites the underlying form /kab/ (a sequence of segments) as the intermediate form /k^{h}ab/, adding the entirely predictable aspiration information to the voiceless stop k. Next, the lengthening rule applies, rewriting the intermediate form /k^{h}ab/ as the surface form [k^{h}aab].

The computation from underlying forms to surface forms is the encoding problem. The input to this computation consists of a grammar g of rewriting rules and an underlying segmental string u. The corresponding output is a surface segmental string $s = g(u)$ where the phonological dependencies represented in g have been encoded into the segmental string u. Therefore, the corresponding decision problem is to decide whether or not a given surface form s encodes the dependencies of given grammar g into a given underlying form u:

Segmental Encoding Problem
Instance: A segmental grammar g and segmental strings u, s
Question: Is s equivalent to $g(u)$?

The computation from surface forms to underlying forms is the decoding problem. The input to this computation consists of a grammar g of rewriting rules and a surface form s.[4] The corresponding output of the segmental decoding computation is the minimal underlying form u that gives rise to $s = g(u)$ according to the grammar g, if such a segmental string exists. Therefore, the corresponding decision problem is to decide whether or not a given surface form has a corresponding underlying form:

Segmental Decoding Problem
Instance: A segmental grammar g and a segmental string s
Question: Is there a segmental string u such that $g(u) = s$?

The segmental model has been without doubt the single most influential phonological theory of this century. Although other phonological models — such as the nonlinear/autosegmental model [5, 27, 16], the prosodic model [17, 18] and

[4] Strict fidelity to the segmental theory of phonology would require us to include a dictionary D of underlying forms as a third input to the decoding problem. The dictionary D lists the underlying phonological forms of all morphemes. These phonological forms (that is, bracketed segmental strings) may be combined by concatenation to form an infinite set D^+ of underlying phonological forms (i.e., the output of the syntax). The complexity class of the segmental encoding and decoding problems is the same with and without the dictionary. Therefore, in order to simplify our presentation, we have chosen not to include the dictionary in our statements of the encoding and decoding problems.

its more recent incarnation as optimality theory [22, 19, 23] — have flourished over the past decade, those models are not able to represent the full range of dependencies attested in the world's phonologies. Therefore, it is misleading to think of these topical phonological models as true alternatives to the segmental model, because they lack the broad range of empirical support enjoyed by the segmental model.

The remainder of this article consists of four sections. In section 1, we present the segmental model in a manner that abstracts the central elements of the theory of segmental phonology as it has developed over the past thirty years. Next, in section 2 we prove that the encoding and decoding problems are undecidable in the segmental model. The vast, unexpected intractability of the segmental model motivates us to undertake a detailed examination of the phonological causes of computational complexity in the segmental model (section 3). Our investigation culminates in a simplified segmental model, that more accurately models the phonological dependencies actually proposed in the phonological literature (section 4). The simplified segmental model is a more restricted and more natural formalization of the segmental theory of phonology. To conclude, in section 5 we prove that the encoding and decoding problems are both NP-complete in this simplified segmental model. Unlike our earlier results [25], the complexity proofs in this article require only a fixed set of *three* phonological features.

1 Segmental Model

In this section we formally specify the segmental model of phonology. Our segmental model is based primarily on the theory presented in *The Sound Pattern of English* (SPE) [4]. This monumental work helped define the field of generative phonology by formalizing central ideas in the field, including the notions of dependency, process, and linguistically-significant generalization. Our model has also been influenced by subsequent work in the segmental framework [8, 9, 13, 26], as well as by the theory of lexical phonology [12, 14, 15, 21, 24].

The segmental model consists of two formal devices: the segment and the rewriting rule. A segment is a set of phonological features that represents an atomic speech sound. Segments are organized into sequences, called segmental strings. A rewriting rule represents an abstract phonological dependency: it specifies a mapping from a given segmental string (that does not encode a particular phonological dependency) to a related segmental string (that does encode that dependency). Let us consider the segmental model in greater detail.

The set of features includes both phonological features, diacritics, and the distinguished feature `segment` that marks constituent boundaries. Phonological features are "abstracted" phonetic features, that may be thought of as representing the idealized state of the vocal tract. In SPE, features may be specified + or - or by an integer value $1, 2, \ldots, N$ where N is the maximal degree of

differentiation permitted for any linguistic feature. The value of N varies from language to language, because languages admit different degrees of differentiation in such features as vowel height, stress, and tone. We simplify the SPE system by restricting features to take natural numbers as values.

Diacritic features are associated with lexical items as a whole; they control the application of rewriting rules. For this reason, the number of diacritic features is bounded above only by the number of rewriting rules in the segmental grammar.[5] An example diacritic is `ablaut`, a feature that marks those stems that must undergo a change in vowel quality, such as tense-conditioned ablaut in the English *sing*, *sang*, *sung* alternation.

Phonological constituents are represented by their boundaries. In particular, the left and right boundaries of a constituent of type κ are marked by a distinguished segment, whose `segment` feature has the value '-' and whose other features encode the constituent type κ. Although SPE provides a detailed feature analysis of the different types of boundaries — formative, word, and sentence boundaries — we will not adopt this somewhat cumbersome aspect of the SPE model.

A set of feature specifications is called a *unit* or sometimes a *segment*. A string of units is called a *matrix* or a *segmental string*.

Rewriting rules formalize the notion of a phonological process, that is, they represent the dependencies implicit in the surface matrices of a given phonology. An *elementary rule* is of the form $ZXAYW \rightarrow ZXBYW$ where A and B may be ϕ or any unit, $A \neq B$; X and Y may be matrices (that is, strings of segments), and Z and W may be thought of a brackets labeled with syntactic categories such as 'S' or 'N' and so forth. Contrary to wide-spread misstatement in the linguistics literature, phonological rules are not technically context-sensitive. Rather, they are unrestricted rewriting rules because they can perform deletions as well as insertions. Common phonological rules include those that insert, delete, alter, or interchange adjacent segments.

The rules are applied in a fixed order to the underlying form, where the input to each rule is the output of it's immediate predecessor. Each rule applies maximally to the current derivation string, that is, simultaneously to all segments in the string. For example, if we apply the rule $A \rightarrow B$ to the string AA, the result is the string BB.

Some phonological processes, such as the assimilation of voicing across morpheme boundaries, are very common across the world's languages. Other processes, such as the arbitrary insertion of consonants or the arbitrary substitution of one segment for another, are extremely rare or entirely unattested. For this reason, all adequate phonological theories must include an explicit measure of

[5] As noted in SPE, "technically speaking, the number of diacritic features should be at least as large as the number of rules in the phonology. Hence, unless there is a bound on the length of a phonology, the set [of features] should be unlimited. There is no point of principle involved here, and to simplify exposition slightly we shall assume the set to be limited by an a priori condition. A similar comment applied to [the set of specifications]." (fn.1, p.390)

the naturalness of a phonological process. A phonological theory must also define a criterion to decide what constitutes two independent phonological processes and what constitutes a legitimate phonological generalization.

Two central hypotheses of segmental phonology are (i) that the most natural grammars contain the fewest symbols and (ii) two rules represent independent phonological processes when they cannot be more compactly combined into a single complex rule [6, 7].

A *complex rule*, then, is a finite schema for generating a (potentially infinite) regular set of elementary rules. To a first approximation, the complex rules of SPE are formed by combining units with the operations of union, concatenation, Kleene star, and exponentiation, using variables whose values range over feature specifications, units, and syntactic categories.[6] Johnson [11] considers a more powerful class of complex rules (and thereby, a different evaluation metric), as well as alternative modes of rule application.[7]

The complex rules are organized into linear sequence $R_1, R_2, \ldots R_n$; they are applied in order to an underlying matrix to obtain a surface matrix. Ignoring a great many issues that are important for linguistic reasons but irrelevant for our purposes, we may think of the derivational process as follows. The input to the derivation is a bracketed string of morphemes (the "underlying form"), the output of the syntax. Recall also that brackets are represented by distinguished boundary segments. The output of the derivation is the "surface form," a string of phonetic units. The derivation consists of a series of cycles. On each cycle, the

[6] Segmental models belong to the class of nonlocal rewriting systems analyzed by Post, rather than to the local systems due to Thue. This is because complex rules can encode a finite number of nonlocal dependencies, and hence the rewriting activity specified by a complex rule can affect parts of the current derivation string (that is, segments in the current intermediate form) separated by an arbitrary distance.

[7] In Johnson's proposal [11], the empty string and each unit are schemata; schema may be combined by the operations of union, intersection, negation, Kleene star, and exponentiation over the set of units. Johnson also uses variables and Boolean conditions in his schemata. This "schema language" is a extremely powerful characterization of the class of regular languages over the alphabet of units; it is not used by practicing phonologists. Because a given complex rule can represent an infinite set of elementary rules, Johnson shows how the iterated, exhaustive application of one complex rule to a given segmental string can "effect virtually any computable mapping," (p.10) that is, *can simulate any TM computation in only one step of the derivation.* Next, he proposes a more restricted "simultaneous" mode of application for complex rules, which is capable of performing at most a finite-state mapping in any single application. The "disjunctive ordering" mode of rule application proposed in SPE is only capable of performing a strictly finite mapping in any single rule application. This mode of application, which is vastly more constrained than either of Johnson's proposals, is also the one used by practicing phonologists. In this article we consider the question of what computations can be performed by a finite set of elementary rules, and hence provide very loose lower bounds for Johnson's unnaturally powerful model. We note in passing, however, that the problem of simply determining whether a given rule is subsumed by one of Johnson's schema is itself wildly intractable, requiring at least exponential space. (The idea of the proof is to construct a complex rule that describes the valid computations of 2^n space-bounded DTM. This rule may be constructed by negating a construction due to Hopcroft and Ullman [10, 350ff].)

ordered sequence of rules is applied to every maximal string of units containing no internal brackets, where each R_{i+1} applies (or doesn't apply) to the result of applying the immediately preceding rule R_i, and so forth. Each complex rule R_i itself generates a disjunctive sequence of elementary rules $R_{i,1}, R_{i,2}, \ldots$ in order of increasing generality.[8] That is, R_i generates a sequence of elementary rules where $R_{i,j}$ precedes $R_{i,k}$ in the sequence iff the preconditions of $R_{i,j}$ subsume the preconditions of $R_{i,k}$; the earliest $R_{i,j}$ that can apply to the current derivation matrix is applied to it, to the exclusion of all other $R_{i,k}$. Each elementary rule applies maximally to the current derivation string, that is, simultaneously to all units in the string. For example, if we apply the rule $A \rightarrow B$ to the string AA, the result is the string BB. At the end of the cycle, the last rule R_n erases the innermost brackets, and then the next cycle begins with the rule R_1. The derivation terminates when all brackets are erased.

In brief, the segmental model consists of a sequence of unrestricted rewriting rules, that can manipulate strings of phonological segments as well as their constituent structure. A derivation step consists of applying all the rewriting rules (in order) to the innermost constituent of the underlying form, and then erasing the outer boundaries of this constituent. A derivation consists of a sequence of derivation steps, that is, the repeated application of the rewriting rules to the underlying form (in order), from the innermost constituent outwards.

We formalize a *segmental grammar* for a given language as containing the following language-particular information: (i) a set F of phonological features and a natural number N representing the maximum number of values a feature may have, (ii) a sequence of complex phonological rules $R = r_1, r_2, \ldots, r_n$, and (iii) a set D of binary diacritic features, $D \subseteq \{d_i : 1 \leq i \leq n\}$, that block the application of individual rules. For example, the rule r_i applies to all underlying forms except those marked by the diacritic $[d_i\ 0]$.

2 Complexity of Segmental Phonology

Let us now consider the question of how much time and space are required to encode and decode phonological dependencies in the segmental model of phonology. The central idea of our complexity analysis is to show how the encoding process can transparently simulate the computation of any arbitrary Turing machine, and therefore both encoding and decoding problems are undecidable. The reader is referred to a text on automata theory, such as the excellent [10], for an introduction to Turing machines.

[8] The interpretation advocated in SPE (p.390ff) is that the true grammar g is the set of elementary rules; the value of this grammar, given by the evaluation metric, is the smallest number of phonological features of any set g' of complex rules that generates the set g. Chapter 9 of Kenstowicz and Kisseberth (1979) contains a less technical summary of the SPE system and a discussion of subsequent modifications and emendations to it.

Lemma 2.1 *The segmental model can simulate the computation of any DTM M on any input w, using only elementary rules.*

Proof. We sketch the simulation. The underlying form x represents the DTM input w, while the surface form y represents the halted state of M on w. The instantaneous description of the machine (tape contents, head position, state symbol) is represented in the string of segments. In particular, each segment represents the contents of exactly one tape square. Special boundary segments {`[-segment]`} are used to mark the rightmost and leftmost edges of the work tape (that is, to represent the fact that all squares outside the boundary segments are blank). The segment representing the currently scanned tape square will also be specified for two additional features, to represent the state of the machine's finite-state control and the direction in which the head will move. Therefore, a total of three features are needed, with a number of feature values determined by the alphabet and finite control of the simulated machine M. Each transition in the transition table δ of the DTM M is simulated by a phonological rule. Four rules are needed to move the head position, and two more rules are required to move the special tape-boundary segment when necessary. Finally, two rules are required erase the entire derivation string when the simulated machine halts, leaving a surface form consisting only of the segment that represents the tape square scanned by the halting state. Thus, a total of $|\delta| + 8$ rewriting rules are used.

There are only two key observations, which do not appear to have been noticed before. The first is that contrary to common misstatement in the linguistics literature, phonological rules are not technically context-sensitive. Rather, they are unrestricted rewriting rules because they can perform deletions as well as insertions. This is essential to the simulation, because it allows intermediate stages of the derivation string to become arbitrarily long. The second observation is that segmental rules can freely manipulate (insert and delete) boundary symbols, and thus the derivation may be prolonged indefinitely: we need only employ a complex rule R_{n-1} at the end of the cycle that adds an extra boundary symbol to each end of the derivation string, unless the simulated machine has halted. The details of the simulation are entirely straightforward, and therefore are not included here. (The reader is referred to the proof of lemma 5.2 below for additional details.) □

The immediate consequences are:

Theorem 1 *Segmental Encoding is undecidable.*

Proof. By reduction from the undecidable problem $w \in L(M)$? of deciding whether a given DTM M accepts an input w. On input instance $\langle M, w \rangle$, the reduction constructs an an underlying form u that represents w, a surface form s that represents the halted, accepting state of the machine M, and a segmental grammar g that simulates the computations of M according to lemma 2.1.

Therefore, $g(u) = s$ if and only if the DTM M halts and accepts on input w. □

Theorem 2 *Segmental Decoding is undecidable.*

Proof. By reduction from the undecidable problem $L(M) = \phi?$ of deciding whether a given Turing machine M accepts any inputs. On input a Turing machine M, the reduction outputs a a surface form s that represents the halted accepting state of the M, and a segmental grammar g that simulates the computations of M according to lemma 2.1. There is an underlying form u that generates s according to the grammar g (ie., $g(u) = s$) if and only if the DTM M halts and accepts some input. □

An important corollary to lemma 2.1 is that segmental grammars form a universal basis for computation. In particular, it is possible to simulate an arbitrary Post tag system using a very simple set of phonological rules. Or we can simulate Minsky's four-symbol seven-state "smallest universal Turing machine" [20] in the segmental model; the resulting grammar contains no more than three features, eight specifications, and 36 trivial elementary rules with less than 250 symbols. These two segmental grammars of universal computation contain significantly fewer symbols than a segmental grammar for any natural language. (By point of comparison, the incomplete SPE grammar for English [4] contains a total of 43 complex rules with more than 500 symbols.) And this is not even the best that can be done. The smallest segmental grammar for the set of all observed segmental strings will be even smaller, because it can take advantage of all computable generalizations among the finite set of observed surface forms, not only the linguistically significant ones. The associated dictionary of underlying forms would represent the Kolmogorov complexity of the observed surface forms with respect to the optimal segmental grammar, that is, the true information content of the observed surface forms with respect to an extremely powerful encoder. In short, even if we ignore questions of feasibility, the smallest segmental grammar capable of enumerating the set of observed surface forms cannot possibly be natural because it must discover too many unnatural generalizations in the finite corpus of observed forms. Therefore, this corollary presents a conceptual and empirical problem for the segmental theory.

In the next section, we examine the phonological causes of the undecidability of segmental computations. We argue that certain very natural design decisions interact in an unnatural manner to give rise to a formal system capable of describing unnatural phonologies.

3 Discussion

The formal specification of the segmental model in section 1 is a straightforward uncontroversial reproduction of textbook segmental phonology [4, 13]. The proof

of the undecidability of the encoding and decoding problems is equally straightforward and should not cause any controversy. Yet the undecidability result itself, which establishes that the segmental model forms a universal basis for computation, cannot help but be controversial. In this section, we examine what empirical facts of phonology lie at the root of segmental undecidability. Our approach will be to identify the formal devices of the segmental model that lead to undecidability, and to summarize their linguistic motivation. We will use those insights in section 4 to simplify the segmental model.

3.1 Phonological Causes of Segmental Undecidability

The proof of lemma 2.1 relies on unrestricted rewriting rules applied in derivations of potentially unbounded length. The rewriting rules of the segmental phonology are unrestricted and can manipulate morpho-syntactic constituent structure. Rules are ordered into a block and interact with each other. In a derivation the block is repeatedly applied, from the innermost constituent out. Repeated rule application may be prolonged indefinitely by means of a phonological readjustment rule that inserts constituent boundaries at the end of each cycle. Let us consider the phonological motivation for each of these devices in turn.

First, phonological dependencies are represented by unrestricted rewriting rules in the segmental model. This is necessary in the segmental model in order to represent irregular morphological forms, as when the underlying forms of the English `go+past` morphemes are mapped to the intermediate phonological form */went/* by a single readjustment rule.[9] In the segmental model, irregular forms therefore demand that the derivation from underlying forms to surface forms be able to substitute one arbitrary segmental string for another. In addition, purely phonological rules are required to insert segments (epenthesis), delete segments (deletion), change the values of segments, and transpose adjacent segments (metathesis). These processes are all most naturally represented by means of unrestricted rewriting rules.

Second, segmental rules may manipulate constituent structure. In the segmental model, constituents are represented by the boundaries of their segmental yield. Recall that boundaries are represented by a distinguished segment specified with the [-`segment`] feature, and that for this reason all segmental rules can manipulate boundaries. Boundaries must be manipulated for two reasons. The first is to reduce the number of cycles in a given derivation by deleting boundaries and flattening syntactic structure, for example to prevent the phonology from assigning too many degrees of stress to a highly-embedded structure. The

[9] In general, readjustment rules map the surface forms given by the syntax into the underlying forms required by the phonology. Thus they map abstract morphemes, such as inflection or agreement, into phonological matrices, and were used in SPE to modify syntactic categories, as when *Fifth Avenue* is mapped from a noun in the syntax to a noun phrase in the phonology, in order that it be assigned the correct stress.

second is to rearrange the boundaries given by the syntax when the intonational phrasing of an utterance does not correspond to its syntactic phrasing (so-called phonological "bracketing paradoxes"). In this case, boundaries are merely moved around, while preserving the total number of boundaries in the string. The only way to accomplish this kind of bracket readjustment in the segmental model is with rules that delete brackets and rules that insert brackets. Boundary insertion rules play a central role in prolonging the derivation indefinitely.

Third, the segmental rules are applied in a fixed order and interact in the derivation. Requiring the segmental rules to be extrinsically ordered results in simpler rule systems, and therefore is preferred on the grounds of simplicity [3, 6, 7]. The English lengthening and flapping processes alluded to the introductory paragraph are best represented as ordered rules that are must be allowed to interact in the derivation.

Fourth, the ordered block of rules is repeatedly applied to the innermost phonological constituent, erasing the innermost boundary markers after each application, stopping only when all boundaries have been erased from the underlying form (the principle of the cycle). This formal device has been used by phonologists; for example, in SPE it is used to account for certain English stress phenomenon.

In short, it would seem that the formal devices of the segmental model are all strongly motivated by empirical facts of phonology. However, we will now argue for a simplified segmental model, that maintains empirical adequacy while taming the intractable interaction of these formal devices.

3.2 Constraining Unnatural Interactions

The simplified segmental model differs from the segmental model of section 1 in several important respects.

The first major difference is that we severely constrain the use of complex rules. Our phonological rules must be elementary, although variables may be used for feature values. Recall that a complex rule is a finite schema for generating a (potentially infinite) set of elementary rules.[10] Because there are only a finite number of possible feature values, each of our simplified phonological rules corresponds to a finite number of elementary rules (arising from all possible specifications of the feature-value variables). This restriction is justified

[10] To a first approximation, the complex rules are formed by combining segments with the operations of union, concatenation, Kleene star, and exponentiation, using variables whose values range over specifications, units, and syntactic categories. The complex rules are organized into linear sequence $R_1, R_2, \ldots R_n$; they are applied in order to an underlying form to obtain a surface form. Each complex rule R_i itself generates a disjunctive sequence of elementary rules $R_{i,1}, R_{i,2}, \ldots$ in order of increasing generality. That is, R_i generates a sequence of elementary rules where $R_{i,j}$ precedes $R_{i,k}$ in the sequence iff the preconditions of $R_{i,j}$ subsume the preconditions of $R_{i,k}$; the earliest $R_{i,j}$ that can apply to the current derivation matrix is applied to it, to the exclusion of all other $R_{i,k}$. See section 1 above for discussion.

because complex rules are used to represent a class of "nonlocal" phonological dependencies that may now be represented using nonlinear representations.

The second major difference is to eliminate the principle of the cycle. There are two reasons for this change. First, cyclic application is an extremely powerful device with slender empirical justification [2]. Second, we are concerned here with the phonology proper, whereas cyclic application relates primarily to the phonology/morphology interface. The immediate consequence of this restriction is to sharply bound the length of segmental derivations.

The third major difference is that we replace the the diacritic features universally-employed by phonologists with the more restricted mechanism of control strings (as explained below). Recall that a diacritic feature is associated with a lexical item as a whole; it controls the application of rewriting rules. The segmental model must allow an unbounded number of diacritic features, simply because it allows an unbounded number of rewriting rules. Each binary segmental feature, whether a diacritic feature or a phonological feature, may be read and written by a segmental rule and so provides the derivation process with a bit of reusable computational space. Therefore, diacritic features are an unbounded source of reusable computational space in the segmental model. Although diacritic features and phonological features are formally equivalent in standard segmental theories, there is an important conceptual distinction to be made between them. Diacritics are properties of segmental strings as a whole, whereas phonological features are properties of individual segments. More importantly, diacritics are used only to control the derivation — they are never affected by it. No phonologist has ever proposed a rule that rewrites the value of a diacritic, although the such rules are entirely natural in the segmental model. Therefore, we eliminate diacritic features altogether.

A fourth difference, which has little effect on the computational complexity of the encoding and decoding problems, is to follow Chomsky and Halle's (1968) suggestion and limit rewriting rules to altering single segments.[11] The only segmental rules required to affect multiple segments are those that transpose adjacent units (metathesis).[12] The first point to be made is that these rules do not apply recursively; a given segment is permuted at most once in the derivation

[11] Chomsky and Halle [4] observe that "a possible direction in which one might look for such an extension of the [segmental] theory is suggested by certain other facts that are not handled with complete adequacy in the present theory. Consider first the manner in which the process of metathesis was treated in Chapter Eight, Section 5. As will be recalled, we were forced there to take advantage of powerful transformational machinery of the sort that is used in the syntax. This increase in the power of the formal devices of phonology did not seem fully justified since it was made only to handle a marginal type of phenomenon. An alternative way to achieve the same results is to introduce a special device which would be interpreted by the conventions on rule application as having the effect of permuting the sequential order of a pair of segments." (p.427)

[12] The segmental theory of [4] required other rules that affected multiple segments (for example, readjustment rules and rules to rewrite irregular forms at the syntax-phonology interface). We accomplish the effects of these rules with other devices, and so have eliminated them from our simplified segmental model.

from underlying forms. Secondly, these rules may be completely eliminated from the segmental grammar for the following reason. A metathesis rule represents the fact that the linear order of two adjacent segments is predictable from the context in which they occur. Recall that the lexicon of underlying forms is the repository of truly random, unpredictable information about sound patterns. Since certain ordering relations between segments are predictable, the lexical entries for underlying forms should not impose a total order on the constituent segments. Rather, an underlying form should be a set of partially ordered segments, while a surface form should be a set of totally ordered segments (that is, a string). A metathesis process, then, ought simply to add an ordering relation between incomparable segments in the partial order; it does not affect the segments themselves, and hence we do not need unrestricted rewriting rules to perform this very simple computation.

A central idea of the segmental approach to phonology is to express all phonological relations using features and concatenation. The approach suggested by metathesis and phonological bracketing paradoxes is to augment the repertoire of phonological relations to include precedence, adjacency, and domination. Such an approach quickly becomes indistinguishable from autosegmental phonology, and so we will not pursue it here.[13]

Let us now formalize our simplified segmental model.

4 Simplified Segmental Model

The simplified segmental model shares much of the essential structure of the segmental model. As in the segmental model, the two central devices are the segment and the rewriting rule. All phonological relations must be represented by segment concatenation, and so constituent boundaries are still represented by means of the distinguished {[-`segment`]} segment.

Rewriting rules represent dependencies implicit in the surface forms of a given language. More precisely, a *phonological rule* is an unrestricted rewriting rule of the form $XAY \rightarrow XBY$ where A and B may be the empty string or any [+`segment`] segment, $A \neq B$, and X and Y are (possibly empty) strings

[13] In particular, the set of phonological features would be limited to articulatory features (advanced, anterior, back, consonental, continuent, distributed, height, nasal, rounded, sonorant, stiff-vocal-cords, strident, spread-glottis). These features organized into the hierarchy given by the feature geometry (coronal, dorsal, labial, laryngeal, peripheral, place, root, soft-palate, stricture, supralaryngeal, root = segment). Segments would in turn be the terminal symbols of two independent hierarchies: the prosodic hierarchy (segment, mora, syllable, foot) and the morphological hierarchy (segment, morpheme, word, phrase). The relations of precedence and adjacency would only be defined on elements of the same type. The left- and right-hand sides of rewriting rules would be defined over the expanded domain of phonological representations. Deletion and epenthesis would then be represented as modification of the segment adjacency relation, while metathesis would be represented as an extension of the segment precedence and adjacency relations. A segmental boundary would be encoded in a rewriting rule as two adjacent nonterminals dominating two adjacent segments.

of segments. (Note that these simplified rewriting rules are unable to affect multiple segments.)

The rewriting rules are applied in a fixed order to the underlying form, where the input to each rule is the output of it's immediate predecessor. Each rule applies maximally to the current derivation string, that is, simultaneously to all segments in the string. For example, if we apply the rule $A \rightarrow B$ to the string AA, the result is the string BB.

A consecutive sequence of rules $r_i, r_{i+1}, \ldots, r_{i+j}$ may be explicitly designated as mutually exclusive. In that case, the application of rule r_k in a given derivation blocks all later rules r_l from applying in that derivation ($i \leq k < l \leq (i+j)$).

The segmental grammar also contains a set $C = \{c_1, c_2, \ldots, c_k\}$ of possible *control strings*, where $C \subseteq \{0, 1, ?\}^n$. Each underlying form is paired with the index i, $1 \leq i \leq k$, of its control string $c_i \in C$. Position i of a control string specifies whether the ith rule should apply ('1'), should not apply ('0'), or should apply optionally ('?') in the derivation involving that underlying form. (Of course, a rule that should apply in the derivation will apply vacuously to a given intermediate form when the left-hand-side of the rule does not match that intermediate form.)

Recall that an underlying form represents the unpredictable information of a given surface form, that a segmental grammar encodes the predictable information of a given language's phonology into phonological representations, and that a surface form is a phonological representation with redundant information (a kind of error-correction, one might speculate). Therefore, we would expect segmental derivations to have the properties of a good encoder and a good decoder, such as being nearly one-to-one and reasonably efficient in both directions. We cannot require the mapping between underlying and surface forms to be strictly one-to-one, because many languages map distinct underlying forms into nondistinct surface forms (as when underlying sound distinctions are collapsed), and every language permits alternate realizations of underlying forms.

Lacking a better understanding of this constraint, we will impose a significantly weaker constraint on segmental derivations, dubbed the constant progression property. The *constant progression property* (CPP) requires that the intermediate forms in a segmental derivation grow progressively more or less by at most a constant amount c, for a progression constant c fixed in advance for each segmental grammar.

To summarize, a *simplified segmental grammar* for a given language contains the following language-particular information: (i) a set F of phonological features and a natural number N representing the maximum number of values a feature may have, (ii) a set $C = \{c_1, c_2, \ldots, c_k\}$ of possible control strings, each $c_i \in \{0, 1, ?\}^n$; and (iii) a sequence of phonological rules $R = r_1, r_2, \ldots, r_n$, some subsequences designated mutually-exclusive, with the progression constant c_p.

5 Complexity of Simplified Segmental Phonology

The encoding problem for the simplified segmental model is: Given two segmental strings u and s, a control string c, and a segmental grammar g, decide whether $g(u, c) = s$. Its counterpart, the decoding problem for the simplified segmental model, is: Given a segmental string s and a segmental grammar g, decide if s can be derived from some underlying segmental string u and control string c according to the grammar g.

Let us begin by establishing the relative complexities of the encoding and decoding problems for the simplified segmental model.

Lemma 5.1 *Simplified Segmental Encoding* $\leq_{\mathcal{P}}$ *Simplified Segmental Decoding.*

Proof. The input to the reduction is instance of Simplified Segmental Encoding: a grammar g, segmental strings u and s, and a control string c. The output of the reduction is an instance of Simplified Segmental Decoding: a new grammar g' and surface form s' such that $\exists u', c'\ [g'(u', c') = s']$ if and only if $g(u, c) = s$. The idea is to construct the grammar g' and surface form s' so that all possible underlying forms u' are ignored by the grammar g'. The most straightforward way to do this is for g' to first erase every segment of the underlying form (by parallel application of the rule $\{\} \rightarrow \epsilon$), insert the segmental string u into the derivation as an intermediate form, and then faithfully apply the rules of g. However, underlying forms are unbounded and so the progression constant of such a grammar must also be unbounded.

Therefore, instead of completely erasing all underlying forms, we will construct the grammar so that only one underlying form can possibly generate the surface form s', and only if $g(u, c) = s$. All other underlying forms will be blocked. To do this, g' begins with a rule $\{\} \rightarrow \{[f\ v]\}$ that specifies every segment for a feature specification $[f\ v]$ that does not appear in the grammar g (and so cannot be erased by the rules of g). The second rule deletes the leftmost segment of the intermediate form. The next $|u|$ rules conspire to insert the segmental string u, one segment at a time, and the remaining rules are exactly the rules of g. The progression constant for g' is the maximum of 1 and the progression constant for g. The only permissible control string is $1^{|u|+2}c$, which makes the first $|u| + 2$ rules of g' obligatory and then does whatever c did with respect to the rules in g. The surface form s' is s. This is an instance of the decoding problem for the simplified segmental model: a segmental string s', a set C' containing one control string, and a grammar g' with a known progression constant.

All underlying forms longer than one segment cannot possibly generate s' in the grammar g' because their segments are all specified with a feature $[f\ v]$ that does not appear on s' and cannot be removed by the grammar g'. Underlying forms of length less than one segment are deleted only to be replaced in the

derivation by the intermediate form u, and then the rules of g are applied to u according to the control string c. Therefore, the grammar g' generates the surface form s' using the one permissible control string if and only if $g(u, c) = s$. □

Lemma 5.2 *Simplified Segmental Encoding is NP-hard.*

Proof. By reduction from the problem of deciding whether a given nondeterministic Turing machine (NTM) M accepts a given input w in polynomial time $p(n)$, $n = |w|$. The reduction outputs a simplified segmental grammar g, segmental strings u and s, and a control string c such that $g(u, c) = s$ if and only if M accepts w in time $p(n)$.

We sketch the simulation. The idea of the proof is to represent the steps of the NTM computation with the intermediate forms of the segmental derivation. The underlying form u represents the NTM input w, while the surface form s represents the halted state of M on w, with the work tape erased. The instantaneous description of the machine (tape contents, head position, and state symbol) is represented in the string of segments. Each segment represents the contents of a single tape square. The segment representing the currently scanned tape square will also represent the state of the machine's finite control and the direction in which the read/write head will move next. Therefore, a total of three features are needed, with a number of feature-values determined by the (size of the) finite control of the simulated machine M.

To be precise, the finite control of the NTM is represented by a distinguished feature f_0 whose possible values are in bijective correspondence with the possible control states, plus the distinguished null value '_' to represent the fact that a given tape square is not currently being scanned. The contents of a tape square (0 or 1) is represented by the feature f_1, and the direction in which the read/write head is about to move (Right, Left, or Stationary) is represented by the feature f_2.

Without loss of generality, let the NTM M be in state q_1 and currently scanning a tape square that contains a '0' value. Furthermore, let M include the transition $\delta(q_1, 0) = \langle q_2, 1, L \rangle$ indicating that when scanning a '0' in state q_1, the machine should write a '1', transition to state q_2, and move the tape head one square to the left. Then our simulation will give rise to the following sequence of intermediate forms,

```
...[_,0,S][_,1,S][q1,0,S][_,0,S]...        ;delta(q1,0)=<q2,1,L>
...[_,0,S][_,1,S][q2,1,L][_,0,S]...        ;move head L
...[_,0,S][q2,1,S][q2,1,L][_,0,S]...       ;finish moving head L
...[_,0,S][q2,1,S][_,1,S][_,0,S]...        ;ready for next transition
```

where [_,0,S] depicts the segment { [f_0 _], [f_1 0], [f_2 S] }.

Each transition of M is simulated by a phonological rule as follows. For every nondeterministic pair of transitions licensed by the NTM transition relation, we

will build two mutually-exclusive phonological rules $r = C \rightarrow C_1$ and $r' = C \rightarrow C_2$, one for each transition. These rules do not look at any context whatsoever (ie., C, C_1, and C_2 are segments). We will construct the dictionary of control strings so that r is optional ('?') and r' is obligatory ('1'). Let the resulting sequence of $|\delta|$ rules be mutually-exclusive. Now we add on four obligatory rules to the sequence, to simulate the rightward or leftward movement of the head position after every transition.

For example, the nondeterministic transition $\delta(q_1, 0) = \{\langle q_2, 1, L\rangle, \langle q_3, 0, R\rangle$ is represented by the following two rules.

`[`q_1`,0,S]` $\rightarrow$ `[`q_2`,1,L]`

`[`q_1`,0,S]` $\rightarrow$ `[`q_3`,,R]`

Similarly, leftward head movement after a transition to state α is accomplished by the following two rules, where α is a variable over the finite set of feature values Q, where Q is state set of the machine M.

`[_,,][`α`,,L]` $\rightarrow$ `[`α`,,][`α`,,L]`

`[`α`,,L][`α`,,]` $\rightarrow$ `[n,,S][`α`,,]`

Rightward head movement is accomplished by a symmetric pair of rules. Notice that the left-hand side of these four rules need only look at two adjacent segments. Let R be the resulting sequence of $|\delta| + 4$ rules, which simulates exactly one step of the NTM computation.

Let the simplified segmental grammar g be $R^{p(n)}$ (that is, the rule sequence R duplicated a polynomial number of times), followed by $p(n)$ obligatory rules $C_h C \rightarrow C_h$, where C_h = `[h,1,S]` is the segment that represents the halted accepting state of the NTM and C is any segment. Together these rules have the effect of erasing the work tape of the NTM if and only if it halts and accepts with the read/write head on the leftmost tape square.) The resulting grammar g has a progression constant of 1.

Now we apply the grammar g to an "underlying" segmental string u of length $p(n)$ that encodes the starting configuration of the NTM, using a unique control string of 1's and ?'s, as constructed above. The lone segment C_h results if and only if the NTM M accepts the input w within time $p(n)$. □

Lemma 5.3 *Simplifed Segmental Decoding is in $\mathcal{NP}$.*

Proof. The decoding problem for the simplified segmental model is in $\mathcal{NP}$ by the following polynomial-time nondeterministic algorithm (sketch). The input to the algorithm is an instance of the decoding problem, that is, a segmental string s and segmental grammar g. The algorithm decides if $\exists u, c\ [g(u, c) = s]$.

Let c_p be the progression constant of g, R the phonological rules of g, and C the possible control strings of g. Let $m = |R|$ and $n = |s|$. By the constant progression property, any underlying form that generates the surface form s contains no more than $n + mc_p$ segments, and no less than $\mathsf{max}(0, n - mc_p)$ segments.

First, the algorithm guesses a control string $c \in C$ and an underlying form u with which to generate s, such that $\mathsf{max}(0, n - mc_p) \geq |u| \geq (n + mc_p)$.

Next, the algorithm applies the rules R to u according to the control string c. If a rule is marked optional ('?'), then the algorithm guesses whether to apply it or not. To apply a rule $r \in R$ to an intermediate form i requires time proportional to $O(|i|\,|r|)$, because in the worst case a rule may be apply to every segment in i. The algorithm terminates when all the rules have been applied to u in order. It accepts if the resulting segmental string equals s; otherwise it rejects. □

Theorem 3 *The encoding and decoding problems for the simplified segmental model are NP-complete.*

Proof. The encoding problem for the simplified segmental model is NP-hard by lemma 5.2, and in $\mathcal{NP}$ by lemmas 5.3 and 5.1. The decoding problem is NP-hard by lemmas 5.2 and 5.1, and directly in $\mathcal{NP}$ by lemma 5.3. □

We note in passing that the NTM simulation described above in lemma 5.2 only requires three phonological features, in addition to the distinguished `segment` feature. These phonological features need only take only a small, fixed number of feature values, because we need only simulate a small, fixed NTM (the smallest NP-complete NTM) in order to establish the NP-hardness of the Segmental Encoding Problem. Similarly, the $\mathcal{NP}$ upper bound of lemma 5.3 holds even if we were to allow an unbounded number of phonological features in the simplified segmental grammar. Thus, allowing an unbounded number of phonological features does not appear to significantly affect the computational complexity of the encoding and decoding problems in either the segmental or the simplified segmental models.

The goal of this article has been to better understand the structure and complexity of phonological dependencies. Our analysis reminds us once again how easily the unbridled interaction of simple computational devices can lead to unnatural extremes of complexity, that must then be tamed by formal constraints.

References

[1] Archangeli, D. *Underspecification in Yawalmani Phonology and Morphology.* PhD thesis, Department of Linguistics and Philosophy, MIT, Cambridge, MA, 1984.

[2] Burzio, L. *Principles of English Stress.* Cambridge University Press, Cambridge, forthcoming.

[3] Chomsky, N. Morphophonemics of modern Hebrew. Master's thesis, University of Pennsylvania, 1951. (Published with revisions, Garland Publishing, New York, 1979).

[4] Chomsky, N., and Halle, M. *The Sound Pattern of English.* Harper & Row, New York, 1968.

[5] Goldsmith, J. *Autosegmental phonology.* PhD thesis, Department of Linguistics and Philosophy, MIT, Cambridge, 1976. (Published in 1979 by Garland Press:New York).

[6] Halle, M. On the role of simplicity in linguistic descriptions. In *Proceedings of Symposia in Applied Mathematics* (1961), pp. 89–94. volume XII, Structure of Language and its Mathematical Aspects.

[7] Halle, M. Phonology in generative grammar. *Word 18*, 1–2 (1962), 54–72.

[8] Halle, M. Tenseness, vowel shift and the phonology of back vowels in modern English. *Linguistic Inquiry 8* (1977), 611–625.

[9] Halle, M., and Mohanan, K. Segmental phonology of Modern English. *Linguistic Inquiry 16*, 1 (1985), 57–116.

[10] Hopcroft, J., and Ullman, J. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[11] Johnson, C. D. *Formal Aspects of Phonological Description.* Mouton, The Hague, 1972.

[12] Kaisse, E., and Shaw, P. On the theory of lexical phonology. *Phonology Yearbook 2* (1985), 85–138.

[13] Kenstowicz, M., and Kisseberth, C. *Generative Phonology.* Academic Press, Orlando, 1979.

[14] Kiparsky, P. Lexical morphology and phonology. In *Linguistics in the Morning Calm*, I. Yang, Ed. Hanshin, Seoul, 1982.

[15] Kiparsky, P. Some consequences of lexical phonology. *Phonology Yearbook 2* (1985), 85–138.

[16] Leben, W. *Suprasegmental Phonology.* PhD thesis, Department of Linguistics and Philosophy, MIT, Cambridge, MA, 1973. (Distributed by the Indiana University Linguistics Club, Bloomington).

[17] McCarthy, J. A prosodic theory of nonconcatenative morphology. *Linguistic Inquiry 12* (1981), 373–418.

[18] McCarthy, J., and Prince, A. Foot and word in prosodic morphology: the Arabic broken plural. *Natural Language and Linguistic Theory 8* (1990), 209–283.

[19] McCarthy, J., and Prince, A. Prosodic morphology i: Constraint interaction and satisfaction. UMass Amherst and Rutgers University, 1993.

[20] Minsky, M. *Computation: finite and infinite machines.* Prentice Hall, Englewood Cliffs, 1969.

[21] Mohanan, K. P. *The Theory of Lexical Phonology.* D. Reidel, Dordrecht, 1986.

[22] Prince, A., and Smolensky, P. Notes on connectionism and harmony theory in linguistics. Technical Report CU-CS-533-91, Department of Computer Science, University of Colorado, Boulder, CO, 1991.

[23] Prince, A., and Smolensky, P. Optimality theory: Constraint interaction in generative grammar. Technical Report 2, Rutgers Center for Cognitive Science, Piscataway, NJ, 1993.

[24] Pulleyblank, D. *Tone in Lexical Phonology.* PhD thesis, Department of Linguistics and Philosophy, MIT, Cambridge, MA, 1983. (Publishedf in 1986 by Reidel:Dordrecht).

[25] Ristad, E. S. Computational structure of generative phonology and its relation to language comprehension. In *Proceedings of the 28th Annual Meeting of the ACL* (Pittsburgh, PA, June 6–9 1990), pp. 235–242.

[26] Rubach, J. Segmental rules of english and cyclic phonology. *Language 60*, 1 (1984), 21–54.

[27] Williams, E. Underying tone in Margi and Igbo. *Linguistic Inquiry 7* (1976), 463–484.