FORMAL ASPECTS OF PHONOLOGICAL DESCRIPTION

by

C. DOUGLAS JOHNSON

University of California, Santa Barbara

1972

MOUTON

THE HAGUE · PARIS

CONTENTS

LIBRARY OF CONGRESS CATALOG CARD NUMBER: 79-190146

Printed in Belgium, by NICI Printers, Ghent

PREFACE

The present work is a revision of my doctoral dissertation, completed in 1970 under the supervision of Professor William S-Y. Wang. The revisions are minor, consisting largely in the correction of typographical errors, occasional rephrasing, and the addition here and there of further illustrative material.

I wish to thank all the members of my dissertation committee for the help they so willingly lent in both intellectual and administrative matters. I am especially grateful to the chairman, Professor Wang, for providing financial support out of National Science Foundation Grant No. GS-1430. I also wish to thank Mrs. Lois Sakamoto for a superb job of typing the original manuscript and my wife Edy for her patience and for the uncountable errands she performed.

University of California, Santa Barbara
August, 1970

C. DOUGLAS JOHNSON

1

INTRODUCTION

The phonology of a language must provide a phonetic representation for each of the infinitely many sentences generated by the syntax. Hence the phonology as well as the syntax is a finite device that accounts for an infinite set of cases. To be sure, the phonology differs in several important respects from the syntax. Where the phrase-structure component takes a single symbol S as initial input and responds with one of an infinite set of alternative outputs, the phonology behaves as a mapping device accepting any of an infinite set inputs and responding in each case with one or at most a small set of alternative outputs. Under current views of syntax, the transformations too constitute a mapping device, though of a radically different sort from the phonology.

A phonological theory must characterize precisely the form of phonologies and the way they process strings. Among the results of such a theory will be a prediction as to what sorts of mappings can be effected by the phonologies of natural languages. Such a prediction would be analogous to a hypothesis in syntax that all natural languages are, say, context-sensitive. Although it has proven difficult to formulate and sustain strong hypotheses of this sort in syntax, we will try to show that in phonology we are somewhat more fortunate.

When confronted with several phonological formalisms with the same mapping capacity, there are several lines of action we could take. We could simply regard the formalisms as empirically equivalent and choose one of them on the basis of practical convenience. The standard view, however, is that formal theories of phonology carry an empirical burden far greater than the mere prediction of phonologically possible mappings and that there are therefore additional and perhaps even more important criteria for choosing among them. In particular, it has been held that the naturalness or plausibility of a phonological process ought to be reflected formally by a corresponding simplicity of formulation. Though this kind of consideration is much less well-defined than mapping capacity, it has played a central role in most discussions of notational devices in phonology and will serve as an important guide to our present investigation.

Our first step will be to examine, in Chapter 2, certain mechanisms for characterizing sets of phonological strings. Although the major focus of our attention will be on how such sets should be represented in the contextual portions of rules, some incidental consideration will be given to the problem of representing morpheme structure. The major result of Chapter 2 will be that the familiar schematic notation formalized in Chomsky and Halle (1968) is quite restricted in its capacity to represent sets of strings and consequently reflects a strong empirical claim concerning the nature of such sets as they occur in phonology. Specifically, we shall see that schemata can represent just the regular sets in the technical sense of automata theory. In Chapter 3 we consider some notational devices which do not add to the representational capacity of schemata but which seem to be necessary for linguistically satisfactory formulations. Most of these devices are already familiar from the literature, but we introduce them systematically to show how they fit into a formal system and to establish the notation to be employed in subsequent chapters. One new departure is suggested: the use of bracket notation to represent set intersection.

It is not until Chapter 4 that rules are formally introduced. There we consider some properties of two diametrically opposed types of rule, the iterative and the simultaneous. Our conclusion is that the iterative type is excessively powerful, being able to effect virtually any computable mapping, while the simultaneous type is highly restricted indeed, being able to effect only the sort of mapping known in automata theory as a finite-state transduction. Thus to confine phonological rules to the simultaneous type is to make another strong claim about phonology, a claim that is essentially correct as far as I can tell.

In Chapter 5 we consider some rule types that are equivalent to the simultaneous in mapping capacity but yield superior formulations in many cases. These new types of rules are called right-linear and left-linear. A body of empirical evidence is considered which leads us to the conclusion that right-linear and left-linear rules should both be allowed in phonological descriptions, although the simultaneous type can apparently be dispensed with.

Linear rules are formalizations of processes which proceed from left-to-right or from right-to-left through a string. Two other ways of formalizing these processes, the restricted iterative and the cyclic, are considered in Chapter 6 and rejected after a review of some empirical evidence.

Distinctive features, originally introduced in Chapter 3, are assumed to be binary up through Chapter 6. In Chapter 7 we consider the effects of allowing integers as feature coefficients. It seems clear that integer coefficients are necessary with at least certain prosodic features. The formal consequence is that certain rules are not strictly finite state and therefore stand as exceptions, albeit of a highly restricted nature, to one of our assertions in Chapter 4. A right-linear tone rule is discussed which manipulates an integrally-valued pitch feature, and it is shown that this rule cannot possibly be formulated with the standard notational devices if simultaneous application is presupposed; a right-linear formulation, on the other hand, seems quite satisfactory. We continue with a discussion of the stress feature, also integrally valued. Our general conclusion is that when stress is a culminative feature, being placed on at most one vowel in any given rule application, then it is either the rightmost or leftmost vowel fitting the structural description of the rule that is affected. Thus in particular we consider unnecessary the complex ordering relations among the subcases of the English Main Stress Rule as given by Chomsky and Halle. It is shown that an alternative formulation, due to Ross, fits quite neatly into the more restricted formalism that we propose.

Certain general conventions to be used throughout should be taken note of. We use $\emptyset$ to designate the null string; thus $X\emptyset Y = XY$ for all strings X and Y. Also, if X is any string then $X^0 = \emptyset$ and $X^i = X^{i-1}X$ for each positive integer i. Thus $X_1 = X$, $X^2 = XX$, $X^3 = XXX$, and so on. It is important to keep in mind that the notational devices just discussed will not be thought of as actually occurring in expressions that appear in phonological descriptions. Rather, they are part of the metalanguage we use to talk about such expressions. This practice is different from that of Chomsky and Halle (1968, Appendix to Chapter 8), who regard $\emptyset$ and superscript and subscript integers as part of the notation of phonological rules.

Certain portions of the text can be skipped without loss of continuity. The beginning and end of such a portion is signaled by (* and *), respectively.

2

SCHEMATA

We will assume that the phonology of any natural language can be described in terms of a fixed universal alphabet of phonological units. For simplicity of exposition we will usually assume that all phonological units are segments, boundary symbols being usually excluded from consideration. A string of phonological units will be called a phonological string.

It is generally accepted that the phonological component of a generative grammar consists wholly or largely of rules which rewrite phonological strings. Each of these rules operates by appropriately altering short substrings (usually single segments) that satisfy certain conditions. Some of these conditions are contextual: a segment will be rewritten in the specified way only if the substring to the left belongs to a certain set (the left environment) and the substring to the right belongs to a certain set (the right environment). Consider, for example, the Sanskrit rule which changes a dental *n* into a retroflex *ṇ* when the *n* is

(a) preceded somewhere in the same word by a retroflex continuant without an intervening palato-alveolar, retroflex, or dental consonant, and
(b) followed immediately by a sonorant.[1]

[1] For other descriptions of the Sanskrit nasal retroflexion rule see Allen (1951), Emeneau and van Nooten (1968: 7), Langendoen (1968: 84), and Whitney (1889: 64-66). Our way of representing vowels and semivowels in underlying forms is similar to that of Zwicky (1965). I wish to thank Professor Murray Emeneau for personally clarifying certain points of Sanskrit grammar. Any errors that remain are entirely my own.

(a) characterizes the left environment of the rule, (b) the right environment. Now suppose that the segments which may appear in phonological representations when this rule applies fall into the classes indicated in the following table.

CONSONANTAL SEGMENTS

	Velar	*Palato-alveolar*	*Retro-flex*	*Dental*	*Labial*	
	k	c	ṭ	t	p	Noncontinuant
	kh	ch	ṭh	th	ph	Noncontinuant
(1)	g	j	ḍ	d	b	Noncontinuant
	gh	jh	ḍh	dh	bh	Noncontinuant
	ŋ	ñ	ṇ	n	m	Noncontinuant
		ś	ṣ	s		Continuant
			r	l		Continuant

NONCONSONANTAL SEGMENTS

a i u h

Then some instances of the left environment of the rule will be

uṣ...	(e.g. *uṣnam* becomes *uṣṇam*)
kṣubhaa...	(e.g. *kṣubhaanaam* becomes *kṣubhaaṇaam*)
draui...	(e.g. *drauinam* becomes *drauiṇam*)

The following strings, however, will not be instances of the left environment:

upaakhiaa...	(e.g. *upaakhiaanaam* remains unchanged)
rathii...	(e.g. *rathiinaam* remains unchanged)

Characterizing a rule environment, then, is a matter of describing a set of phonological strings. How such sets are to be represented formally is a crucial question of phonological theory.

In the past it has been proposed that a phonological description also contain a set of statements or rules which characterize the set of phonologically admissible morphemes in a language. The status and organization of this morpheme-structure component, as we shall call it, is a matter of some controversy, but if we grant it some sort of existence we are again faced with the problem of representing a set of strings, namely, those strings that are phonologically admissible as morpheme shapes. (On the other hand, there seems to be no need at all for a special component to describe the set of admissible phonetic strings, since this set is determined indirectly by the morpheme-structure component and the phonological rules).

Several possible mechanisms for representing sets of phonological strings immediately suggest themselves. We might, for example, generate such sets by means of phrase-structure or even transformational grammars. A number of authors (Romeo 1964, Waratomasikkhadit 1964) have proposed that the admissible phonetic or phonemic strings of a language be described in this manner. However, no one to my knowledge has suggested the use of such powerful devices to characterize the environment of a phonological rule. Morpheme structure too has usually been described otherwise by generative phonologists.

Another approach would be to allow the use of string variables and truth-functional conditions. The left environment of the Sanskrit nasal retroflexion rule could then be given as

PAQ

Conditions: A is a retroflex continuant;
$Q \neq RBS$ if B is a palato-alveolar, retroflex or dental consonant.

This way of describing environments is often found in the literature, though usually conjunction with other formal devices and not in the pure form exemplified here.

Most formalisms actually used or proposed for writing phonological rules and representing morpheme structure appear to be versions of a schematic notation involving two fundamental devices:

(2) (a) Explicit finite lists. The usual notation is $\{X_1, \ldots, X_n\}$, where $X_1, \ldots, X_n$ are the listed items.

(b) Some means of representing an infinite list of the form $X^0, X^1, X^2, X^3, \ldots$. Both $(X)_0$ and $(X)^*$ have been used in this function. We will use $(X)^*$.

Using this notation we can represent the left environment of the Sanskrit rule by the expression

(3) $(\{A_1, \ldots, A_q\})^* \{r, ṣ\} (\{k, kh, g, gh, ŋ, p, ph, b, bh, m, a, i, u, h\})^*$

where $A_1, \ldots, A_q$ are all the segments of table (1).

Any meaningful expression involving brace and star notation will be called a schema. From the informal explanation just given it is probably fairly easy to construct and interpret the schemata needed in phonology. However, we will not leave the matter to the reader's intuition but proceed to a formal development. To begin with, we characterize well-formed schemata recursively in (4). The term "elementary symbol", used in (4), is for present purposes equivalent to "phonological unit".

(4) (a) $\emptyset$ is a schema and each elementary symbol is a schema.
(b) If $X_1, \ldots, X_n$ are schemata then so is $X_1 \ldots X_n$.
(c) If $X_1, \ldots, X_n$ are schemata then so is $\{X_1, \ldots, X_n\}$.
(d) If X is a schema then so is $(X)^*$.

To verify that (3), for example, is a schema, we first note that since each segment is a schema by virtue of (4a), the following are also schemata by virtue of (4c):

$\{A_1, \ldots, A_q\}$
$\{r, ṣ\}$
$\{k, kh, g, gh, ŋ, p, ph, b, bh, m, a, i, u, h\}$

Then because of (4d) each of the following is a schema too

$(\{A_1, \ldots, A_q\})^*$
$(\{k, kh, g, gh, ŋ, p, ph, b, bh, m, a, i, u, h\})^*$

That (3) is a schema now follows directly by virtue of (4b).

At this point it would be well to establish explicitly certain notational conventions which we have in fact been tacitly observing.

Late capital letters U, ..., Z range over arbitrary schemata, the earlier letters P, ..., T over elementary strings (strings of elementary symbols), and the very early letters A, ..., E over elementary symbols.

The interpretation of schemata in phonology is usually given in the form of conventions which expand schemata into explicit lists, which may be finite or infinite. The standard conventions for brace and star are given in (5).

(5) (a) $X\{Y_1, ..., Y_n\}Z$ expands to the finite list $XY_1Z, ..., XY_nZ$
(b) $X(Y)^*Z$ expands to the infinite list $XY^0Z, XY^1Z, XY^2Z, XY^3Z, ...$

(cf. Chomsky and Halle, 1968: 394, 398). By recursively applying these conventions one presumably ends up with a list of elementary strings. These strings constitute the set represented by the original schema.

In general, of course, we cannot literally write out lists resulting from expansions since these may very well be infinite. (5) is apparently just a metaphorical version of (6).

(6) (a) Each elementary string represents the set consisting of that string alone.
(b) $X\{Y_1, ..., Y_n\}Z$ represents the union of the sets represented by $XY_1Z, ..., XY_nZ$.
(c) $X(Y)^*Z$ represents the union of the sets represented by $XY^0Z, XY^1Z, XY^2Z, XY^3Z, ...$

It is convenient to say that a string is subsumed under a schema if it belongs to the set represented by that schema.

Alternatively, then, we state (6) as in (7).

(7) (a) Each elementary string subsumes itself and only itself.
(b) P is subsumed under $X\{Y_1, ..., Y_n\}Z$ if and only if it is subsumed under XY_iZ for some i, $1 \leq i \leq n$.
(c) P is subsumed under $X(Y)^*Z$ if and only if it is subsumed under XY^iZ for some $i \geq 0$.

Another way of interpreting (5) is this. Suppose that instead of replacing a schema by an expanded list we replace the schema by an arbitrarily chosen item in that list. Continuing in this way we eventually arrive at a particular elementary string. Suppose we say that the sequence of items written down in this process is a chain, and that adjacent items constitute links. We can formalize this alternative interpretation of (5) as in (8).

(8) (a) A brace-removing link is an ordered pair of the form $(X\{Y_1, \ldots, Y_n\}Z, XY_iZ)$ where $1 \leq i \leq n$.
(b) A star-removing link is an ordered pair of the form $(X(Y)^*Z, XY^iZ)$ where $i \geq 0$.
(c) A link is an ordered pair that is either a brace-removing link or a star-removing link.
(d) A chain is a nonempty sequence $(X_1, \ldots, X_n)$ in which (X_i, X_{i+1}) is a link for each i, $1 \leq i \leq n-1$.

Under this conception a schema X represents the set of all P which occur at the end of chains beginning with X. An example of a chain beginning with schema (3) and ending in the string *brahma* is given in (9). For brevity we have set

$U = \{A_1, \ldots, A_q\}$
$V = \{k, kh, g, gh, ŋ, p, ph, b, bh, m, a, i, u, h\}$.

For each line we indicate whether it forms a brace-removing link or a star-removing link with its predecessor.

(9)	(U)*{r, ṣ} (V)*	
	U {r, ṣ} (V)*	star-removing
	b {r, ṣ} (V)*	brace-removing
	br(V)*	brace-removing
	brVVVV	star-removing
	braVVV	brace-removing
	brahVV	brace-removing
	brahmV	brace-removing
	brahma	brace-removing

Some fairly obvious consequences of the definitions given so far are stated in (10).

(10) (a) (X, X′) is a link if and only if there are schemata W, Y, Y′, Z such that X = WYZ, X′ = WY′Z, and (Y, Y′) is a link.
(b) If (X_1, ..., X_n) and (X_n, ..., X_{n+m}) are chains, then so is (X_1, ..., X_n, ..., X_{n+m}).
(c) If $P \neq X$, then P is subsumed under X if and only if there is a link (X, X′) such that P is subsumed under X′.

The question of primary interest, of course, is whether or not (7) and (8) yield equivalent interpretations of schemata. The answer is yes; that is,

(11) P is subsumed under X if and only if some chain begins with X and ends in P.

(*First we prove (12)).

(12) If there is a chain beginning with X and ending in P, then X subsumes P.

Let the chain be (X_1, ..., X_n). Suppose n = 1. Then $X_1 = X_n = P$. But by (7a) P subsumes itself. Suppose next that $n > 1$ and that (12) is true for all chains of length less than n. By the inductive hypothesis P is subsumed under X_2 and hence, because (X_1, X_2) is a link, under X_1 (cf. (10c)).

To prove the converse of (12) we make use of a depth measure. The depth of any schema X, denoted d(X), is defined recursively as follows:

(13) (a) If X is an elementary string then d(X) = 1.
(b) If X = YZ then d(X) is the maximum of d(Y), d(Z).
(c) If X = {Y_1, ..., Y_n} then d(X) is one greater than the maximum of d(Y_1), ..., d(Y_n).
(d) If X = (Y)* then d(X) is one greater than d(Y).

This definition assigns a unique depth to every schema and is so framed that every schema is at least as deep as any schema it contains.

Schema (3) provides convenient examples. Letting U and V have the same values as in (9) we have:

Expression	*Depth*
any segment	1
brahma	1
{r, ṣ}	2
U	2
V	2
U {r, ṣ}	2
{r, s} V	2
(U)*	3
(V)*	3
(U)* {r, ṣ}	3
{r, ṣ} (V)*	3
(U)* (V)*	3
(U)* {r, ṣ} (V)*	3

For each schema X of depth greater than 1 there will be schemata W, Y, and X such that

(i) X = WYZ;
(ii) Y has the form {...} or (...)*; and
(iii) Y has the same depth as X.

The number of distinct triples (W, Y Z) satisfying (i)-(iii) will be called the linear complexity of X, denoted lc (X). Thus if X is schema (3), X will have a linear complexity of 2 because both of the following triples, but no others, satisfy (i)-(iii):

(∅, (U)*, {r, ṣ} (V)*)
((U)* {r, ṣ}, (V)*, ∅)

where U and V are as in (9). Now suppose that P is subsumed under X and that (W, Y, Z) is a triple satisfying (i)-(iii). Because of (10a) and (10c) there is a schema Y′ such that (Y, Y′) is a link and P is subsumed under WY′Z. The schema Y′ must be shallower than Y and hence shallower than X (cf. (13c) and (13d)), while Y itself is as deep as X because by hypothesis it satisfies (iii) above.

Consequently, if the linear complexity of X is greater than 1, the linear complexity of WY′Z will be one less than that of X, although the depths of X and WY′Z will be the same. On the other hand, if the linear complexity of X is 1, the depth of WY′Z will be one less than that of X.

We can now show (14)

(14) If the depth of X is greater than 1 and P is subsumed under X, then there is a chain beginning with X and ending in some schema which is shallower than X and which subsumes P.

Suppose first that $1c\ (X) = 1$. It follows from the remarks in the preceding paragraph that there is a link (X, X′) where X′ is shallower than X and subsumes P. But this link is also a chain. Now suppose that (14) is true for $1c(X) \leq k$. Consider the case where $1c\ (X) = k + 1$. It follows from the remarks of the preceding paragraph that there is a link (X, X′) where X′ has the same depth as X, has a linear complexity less than that of X, and subsumes P. By the inductive hypothesis there is a chain $(X', X_1, \ldots, X_n)$ such that X_n is shallower than X′ and subsumes P. But $(X, X', X_1, \ldots, X_n)$ is also a chain, in fact, a chain that begins with X and ends in a schema that is shallower than X and subsumes P.

We are now ready to consider the converse of (12) directly. This is given in (15).

(15) If P is subsumed under X, then there is some chain beginning with X and ending in P.

Suppose $d(X) = 1$. Then $P = X$, and the sequence consisting of P alone is a chain beginning with X and ending in P. Suppose next that (15) is true for $d(X) \leq k$. Consider the case where $d(X) = k + 1$. By (14) there is a chain $(X, X_1, \ldots, X_n)$ in which X_n is shallower than X and X_n subsumes P. By the inductive hypothesis there is a chain $(X_n, \ldots, X_{n+m})$ where $X_{n+m} = P$. But $(X, X_1, \ldots, X_{n+m})$ is a chain beginning with X and ending in P. This concludes the proof of (11).*)

A third way of interpreting schemata, the last we shall consider, is given in (16).

(16) (a) $\emptyset$ subsumes itself and only itself, and each elementary symbol subsumes itself and only itself.

(b) P is subsumed under XY if and only if there are strings Q and R such that P = QR, Q is subsumed under X, and R is subsumed under Y.

(c) P is subsumed under $\{X_1, ..., X_n\}$ if and only if it is subsumed under some X_i ($1 \leq i \leq n$).

(d) P is subsumed under $(X)^*$ if and only if it is subsumed under some schema of the form X^i ($i \geq 0$).

(16) interprets schemata in precisely the same way as (7) and (8). (*To demonstrate this we need only show that (16) implies (7) and conversely. That (16) implies (7) can be seen from the fact that (7a), (7b) and (7c) can be derived from (16) by taking (16a), (16c), and (16d), respectively, in conjunction with (16b). That (7) implies (16) can be seen from the following considerations. (16a), (16c) and (16d) are just special cases of (7a), (7b), and (7c), respectively. As to (16b), we can reason as follows. First, suppose we know that P = QR, where Q is subsumed under X and R is subsumed under Y. Then there will be chains $(X_1, ..., X_n)$ and $(Y_1, ..., Y_m)$ such that $X = X_1$, $X_n = Q$, $Y = Y_1$, and $Y_n = R$. However, $(X_1Y_1, ..., X_nY_1, ..., X_nY_m)$ is also a chain (cf. (10a)); in fact, it is a chain beginning with XY and ending in P. On the other hand, suppose we know that P is subsumed under XY. Then there will be a chain beginning with XY and ending in P, and this chain must have the form $(X_1Y_1, ..., X_nY_n)$ where $X = X_1$, $Y = Y_1$, and for each i, $1 \leq i \leq n-1$, either

(i) (X_i, X_{i+1}) is a link and $Y_i = Y_{i+1}$, or else
(ii) $X_i = X_{i+1}$ and (Y_i, Y_{i+1}) is a link.

(Again, cf. (10a).) Since $(X_1, ..., X_n)$ and $(Y_1, ..., Y_m)$ can be turned into chains merely by deleting repetitions, $X_1 = X$ subsumes X_n and $Y_1 = Y$ subsumes Y_n. But $P = X_nY_n$, so that we can take X_n and Y_n, respectively, as Q and R.*)

The result just obtained is of some significance because it means that schemata are just notational variants of regular expressions, familiar from theory of finite automata (cf. the definitions in McNaughton and Yamada 1960, where the term 'denote' corresponds to our 'subsume' and where the notation $X_1 \cup \ldots \cup X_n$ corresponds to our $\{X_1, \ldots, X_n\}$). The only discrepancy is the trivial one that we have no method of representing the null set, a gap we can immediately remedy by introducing the symbol O (to be distinguished from $\emptyset$ denoting the empty string in our metalanguage). We can, then, represent with schemata all and only those sets that we can represent with regular expressions. These sets, which are said to be regular, constitute a highly restricted family, ranking lowest in the following familiar hierarchy:

Recursively enumerable sets
Context-sensitive sets
Context-free sets
Regular sets.

As is well known, each family of sets in this list is a proper subfamily of any family listed above it.

To propose, then, that schematic notation is adequate for writing phonological descriptions is to assert that all the sets that need to be referred to are regular. This is a strong claim in view of the highly restricted nature of regular sets. If the claim is correct, as I believe it is by and large, phonology stands in sharp contrast to syntax, where comparably strong hypotheses seem not to be tenable. Furthermore, we now have a profound reason for rejecting the proposal discussed earlier that phrase-structure or transformational grammars be used to generate sets of phonological strings, for many grammars of these types generate nonregular sets (indeed, some recent work of Kimball (1967), of Peters and Ritchie (1969) and of Ginsburg and Partee (1969) suggests that we can generate any recursively enumerable set with a transformational grammar). The free use of string variables is also excluded by the regularity claim, for with such variables we can generate such sets as (i) and (ii), which are known not to be regular.

(i) The set of all strings of the form PAP.
(ii) The set of all strings of the form PAQ, where $P \neq Q$.

To conclude this chapter, let us consider which of the three equivalent ways of interpreting schemata should be taken as axiomatic. This question is of no great theoretical import, but it must be decided before we proceed to further development of the formalism in the next chapter. In fact, we will find it most convenient to take (16) as axiomatic, and we will henceforth refer to clauses (16a)-(16d) as interpretive axioms. (8) then becomes a set of auxiliary definitions and (7) becomes a set of theorems along with the other assertions made in this section about subsumption. The clauses (4a) through (4d), which define well-formed schemata, will be known as constructive axioms.

3

REFINEMENTS OF THE FORMALISM

Neither braces nor star can be removed from the formalism of schemata without drastically reducing the family of representable sets. Hence both braces and star must be regarded as primitive notational devices. However, we can exclude from the primitive notation every expression of the form $X \{Y_1, \ldots, X_n\} Z$ in which X or Z is nonnull, since such an expression represents exactly the same set as $\{XY_1Z, \ldots, XY_nZ\}$. For an example we can turn again to schema (3) of the preceding section. Letting U and V be as in (9), we can write this schema as $(U)^* \{r, ṣ\} (V)^*$. This schema, which is not primitive, represents the same set as the primitive schema

$$\{(U)^*r(V)^*, (U)^*ṣ(V)^*\}$$

Although we can represent all phonologically relevant sets with primitive schemata, assuming these sets to be regular, we cannot always do so in a way that is satisfactory for linguistic purposes, because we frequently cannot express linguistically significant generalizations concerning the subsumed strings. For this reason it is essential to introduce nonprimitive devices of an abbreviatory nature into the formalism. The classic example of such a device is nonprimitive brace notation, as exemplified in schema (3). Since this use of braces is so well established in phonology, we will accept it here without further comment.

Another aspect of our formalism that needs clarification is the status of phonological units. We will make the usual assumption that it is not the alphabet of units that is to be taken as primitive but rather some fixed finite universal set of distinctive features F_1, ..., F_d, each of which is associated with a set of possible coefficients. Phonological units are then defined by the constructive axiom (17).

(17) Each expression of the form $[K_1F_1, ..., K_dF_d]$, where K_i is a possible coefficient of F_i, is a phonological unit.

The expressions K_iF_i will be called feature specifications. Of course, since certain combinations of features specifications are inherently impossible, (17) can be regarded only as a first approximation to the definition of unit. However, we will attempt no further refinement here.

The distinctive features to be assumed here unless otherwise noted will be those proposed by Chomsky and Halle (1968: Chapter 7) as amended on page 354 to include a feature of syllabicity. We also assume that the possible coefficients of each feature constitute a finite set; in fact, we generally take these coefficients to be just plus and minus. Under these assumptions the alphabet of units is still finite, and the hypothesis of regularity put forward in the previous chapter remains unaffected. The consequences of allowing an infinite set of coefficients for some feature (all the positive integers, say), which may involve a departure from regularity, will be considered in a later chapter.

The distinctive features form the basis of a highly important system of abbreviatory notation. A simple version of this notation is given by (18), which includes one constructive axiom (18a) and one interpretive axiom (18b).

(18) (a) Each feature specification is a schema.
(b) P is subsumed under a feature specification if and only if P is a unit containing that specification.

Thus *t* is subsumed under +cor(onal) but not under −cor.

Suppose now we extend the formalism to include the new axioms (19a) and (19b), which are constructive and interpretive, respectively.

(19) (a) If $X_1, ..., X_n$ are schemata, then $[X_1, ..., X_n]$ is a schema.

(b) P is subsumed under $[X_1, ..., X_n]$ if and only if it is subsumed under each X_i.

In other words, $[X_1, ..., X_n]$ represents the intersection of the sets represented by $X_1, ..., X_n$. Since the intersection of two or more regular sets is regular, the bracket notation hereby introduced does not increase the family of representable sets and can therefore be regarded as nonprimitive.

The main justification for (19) is that it yields the full apparatus of distinctive feature notation when conjoined with (18). Thus *t* is subsumed under [+cor, −voice] because it is subsumed under +cor and −voice, but is not subsumed under [+cor, +voice] because it is not subsumed under +voice. To consider a slightly more complex example possible under the extended formalism, *a* is subsumed under [+syl, {+tns, −high}] because it is subsumed under both +syl and {+tns, −high}, and it is subsumed under {+tns, −high} because it is subsumed under −high. Two minor differences between our notation and the customary one should be noted. First, we allow a feature specification to stand unbracketed to represent the set of all units containing that specification. However, since [X] is equivalent to X, we can if we wish bracket otherwise unadorned feature specifications, a practice we will follow. The other difference is that our definitions imply [ ] = [0]; hence, [ ] stands for the set containing just the null string rather than the set of all phonological units. For the latter set we will use the special symbol $.

The need for distinctive feature notation is glaringly apparent in schema (3), and we are now able to recast it more adequately. Suppose that the segments of table (1) are specified in part as follows (this analysis is based on Chomsky and Halle 1968: 314):

	Coronal	*Distributed*	*Anterior*	*Continuant*
k, kh, g, gh, ŋ	−		−	−
c, ch, j, jh, ñ	+	+	−	−
ṭ, ṭh, ḍ, ḍh, ṇ	+	−	−	−
t, th, d, dh, n	+	−	+	−
p, ph, b, bh, m	−		+	−

ś	+	+	−	+
ṣ	+	−	−	+
s	+	−	+	+
r	+	−	−	+
l	+	−	+	+
a, i, u, h	−		−	+

Schema (3) can then be reformulated as:

($)*[+cont, +cor, −ant, −distr] ([−cor])*

Although bracket notation has been motivated chiefly by its utility in representing classes of phonological units, it has, under our formalism, a potentially much vaster range of application. Consider, for example, the readjustment rule of English (Chomsky and Halle 1968: 175) which marks a vowel as exempt from part of a certain laxing rule if that vowel is followed by a string subsumed under

[+cons, +ant, +cor] [+cons, +cor]

To be subsumed under this schema a string must have the following characteristics:

(i) it must consist of two segments,
(ii) it must consist of consonantal coronal segments, and
(iii) it must begin with an anterior segment.

Notice, however, that the schema does not directly express these three generalizations; rather, it describes each of the two segments in the cluster separately. It might be held that this is undesirable. If so, we can use bracket notation to express (i)-(iii) directly, as follows:

[$$, ([+cons, +cor])*, [+ant]$]

Whether this is to be regarded as an improvement is disputable, of course. Consider another case. It has become customary to use an expression of the form X_1 to stand for $X^1(X)^*$. Thus $[-\text{syl}]_2$ would subsume clusters of two or more nonsyllabics. We can reproduce the essence of this subscript notation by means of brackets and \$ if we want; for example, we can write [([−syl])*, \$\$(\$)*]. This expression says directly that (i) the subsumed strings consist entirely of nonsyllabics and (ii) the subsumed strings consist of two or more units. The bracket and \$ notation may in some respects be superior to the subscript notation. Consider the following phenomenon, discussed by Kiparsky (1968: 180). Old English had a rule that laxed a vowel before a cluster of at least three consonants, and Early Middle English generalized the environment to include clusters of two consonants as well. Without subscripts or extended use of brackets we could describe this change as simplification of the schema [−syl] [−syl] [−syl] to [−syl] [−syl]. However, a rule using this schema does not directly express the generalization that members of the clusters are all consonants; the segments are specified independently. We might then want to write the schemata of the two historic versions of the rule as $[-\text{syl}]_3$ and $[-\text{syl}]_2$. Then, however, if the subscripts are an official part of the notation there is no difference in simplicity, and one way of explaining the historic change in the rule is lost. It would seem that the schemata [([−syl])*, \$\$\$] and [([−syl])*, \$\$] would resolve this dilemma if \$ were assigned some small fractional cost less than that of a distinctive feature.

The examples just discussed are intended only as illustrative. I have not pursued a thorough investigation of bracket notation, and it is entirely unclear to me how far its use should be extended. Henceforth, therefore, we will usually confine our use of brackets to the representation of classes of phonological units in conformity with the usual custom.

Still another abbreviatory device that has come into wide use is the variable. In the preceding section we saw that variables ranging freely over strings make it possible to represent nonregular sets, and since we are assuming that nonregular sets do not come up in phonology, we will want to constrain variable notation in some appropriate way. What we will do is allow variables to range only over feature coefficients and phonological units.

The basic formalism for variable notation is given in (20). We have adopted the convention of using capital letters from G through

O for strings that are not necessarily either schemata or phonological strings.

(20) (a) If K is a coefficient (resp.: unit) and L is a coefficient (resp.: unit) variable, then K is a substitution instance of L.
(b) If GKH is a schema and K is a substitution instance of L, then GLH is a schema.
(c) Let X be a schema of the form $G_0L_1G_1 \ldots L_nG_n$, where each L_i is a variable and each G_i is free of variables. A substitution instance of X is a schema of the form $G_0K_1G_1 \ldots K_nG_n$, where K_i is a substitution instance of L_i and $K_i = K_j$ if $L_i = L_j$.
(d) If X contains variables, P is subsumed under X if and only if it is subsumed under a substitution instance of X.

In deference to custom, but not from any consideration of principle, we will use early Greek letters α, β, γ, ... as coefficient variables. Late Greek letters σ, τ, υ, ... will be used as unit variables. Actually, unit variables have appeared in a different guise under another name, but we will not be ready to discuss this matter until we treat rule formalism.

As the formalism now stands it is inconsistent because paradoxes such as the following arise at every turn. According to (20) both *t* and *a* are substitution instances of σ and hence both are subsumed under σ. Also, according to (20), the strings *tt* and *aa*, but not *ta* or *at*, are subsumed under $\sigma\sigma$. But by (16b) both *ta* and *at* are subsumed under $\sigma\sigma$. To avoid this contradiction we must replace (16b) by (21).

(21) If neither X nor Y contain variables, then P is subsumed under XY if and only if there are strings Q and R such that P = QR, Q is subsumed under X, and R is subsumed under Y.

Now for a simple example involving unit variables. The set of all strings of the form ABBA, where A is a vowel and B a consonant, can be represented by the schema

$[+\text{syl}, \sigma]\ [-\text{syl}, \tau]\ [-\text{syl}, \tau]\ [+\text{syl}, \sigma]$

Replacing the variables by *a* and/or *t* in all the allowable ways, we obtain the following as substitution instances of the above schema:

$[+\text{syl, a}]\ [-\text{syl, t}]\ [-\text{syl, t}]\ [+\text{syl, a}]$
$[+\text{syl, a}]\ [-\text{syl, a}]\ [-\text{syl, a}]\ [+\text{syl, a}]$
$[+\text{syl, t}]\ [-\text{syl, t}]\ [-\text{syl, t}]\ [+\text{syl, t}]$
$[+\text{syl, t}]\ [-\text{syl, a}]\ [-\text{syl, a}]\ [+\text{syl, t}]$

The first of these substitution instances subsumes only the string *atta*. The remaining three subsume nothing whatever, because nothing can be subsumed under [−syl, a] or [+syl, t]. This is precisely the desired result.

There are certain auxiliary devices which are of little use by themselves but can be employed to good effect in conjunction with variable notation. Among these are conditions, counterparts, and coefficient strings.

Conditions are frequently used to further restrict the ways in which variables can be replaced. Let us say that an expression of the form (K = L), where K and L are both units or both coefficients is an atomic condition. Nonatomic conditions can be constructed by using negation, denoted by a prefixed hyphen, or any of the sentential connectives 'and', 'or', 'if...then'. To bring conditions into schemata we add (22) to the existing axioms. (22a) is constructive, (22b) interpretive.

(22) (a) If X is a schema containing no condition and if K is a condition, then X:K is a schema.
(b) P is subsumed under X:K if and only if P is subsumed under X and K is true.

Thus X:K represents either the same set as X or the null set, according as K is true or false. Consider, for example, the schema $\sigma\tau{:}-(\sigma = \tau)$, which represents the set of all clusters consisting of two nonidentical units. Among the substitution instances of this schema are $nt{:}-(n = t)$ and $tt{:}-(t = t)$. $nt{:}-(\text{n} = \text{t})$ has a true condition, *n* and *t* being in fact nonidentical; consequently it represents the same set as *nt* (which subsumes only itself). *tt*: $-(t = t)$ has a false condition and subsumes nothing.

Schemata that might be referred to as counterpart expressions are introduced in (23).

(23) (a) If $X_1, \ldots, X_n$ are feature specifications and A is a phonological unit, then $[X_1, \ldots, X_n, (A)]$ is a schema.

(b) A is subsumed under
$[X_1, \ldots, X_p, KF_i, ([Y_1, \ldots, Y_q, LF_i, Z_1, \ldots, Z_r])]$
if and only if it is subsumed under
$[X_1, \ldots, X_p, ([Y_1, \ldots, Y_q, KF_i, Z_1, \ldots, Z_r])]$.

(c) $[(A)]$ subsumes A and only A.

As an example, consider the counterpart expression [−voice, (d)]. (23b) tells us that to interpret this expression we should substitute −voice for the feature specification +voice in *d*; the result will be the voiceless counterpart of *d*, which is *t*. The primary use of counterpart expressions will be in the structural change portion of rules, treated in the next chapter.

Coefficient variables become more flexible instruments if strings of coefficients are allowed to occur in feature specifications and if the familiar equivalences $++ = +$, $-- = +$, $+- = -$, and $-+ = -$ are adopted as interpretive principles. We introduce this commonplace notation in (24).

(24) (a) If K is a nonempty string of coefficients of F_i, then KF_i is a schema.

(b) Let K be a possibly empty string of coefficients of F_i. Then P is subsumed under $++KF_i$ (resp.: $--KF_i$, $+-KF_i$, $-+KF_i$) if and only if it is subsumed under $+KF_i$ (resp.: $+KF_i$, $-KF_i$, $-KF_i$).

The variable notation and associated auxiliary devices just introduced do not increase the family of sets representable by schemata. It is obvious that this is true of the auxiliary devices considered by themselves; for the very axioms that interpret them are essentially statements of equivalence between schemata that contain them and schemata that do not; thus X:K is equivalent to X or $\varnothing$, each counterpart expression is equivalent to a specific phonological unit, and each schema of the form KF_i is equivalent to either $+F_i$ or $-F_i$. As to unit and coefficient variables, each of them has only a finite set of substitution instances. Hence any schema X containing such variables, but no other variables, will have a finite set of substitution instances $Y_1, \ldots, Y_n$ and will be equivalent, therefore, to the variable-free schema $\{Y_1, \ldots, Y_n\}$.

The schematic notation as it now stands will form the basis of our discussion of rules, to which we turn in the next chapter. Some customary devices that have not so far been mentioned, such as angle brackets and parentheses, will be discussed in Chapter 7.

Some conventions adopted for solely typographical reasons are the following. A list of items separated by commas, whether enclosed in brackets or braces, will frequently be displayed vertically. In addition, we will write X* instead of (X)* where X is a braced or bracketed schema or \$. Thus our final reformulation of schema (3) is

$$\$^* \begin{bmatrix} +\text{cont} \\ +\text{cor} \\ -\text{ant} \\ -\text{distr} \end{bmatrix} [-\text{cor}]^*$$

4

ITERATIVE AND SIMULTANEOUS RULES

A generative phonology is a system of rules for mapping phonological strings into phonetic realizations. We will suppose that the most rudimentary form such a rule can have is $P \rightarrow P'/Q - R$, where P, Q, R, and P′ are phonological strings. This type of rule will be referred to as elementary and will be said to have QPR as its input and QP′R as its output. Usually such a rule is thought of as applying to any string of the form SQPRT, but for convenience in formalization we will take the more atomistic view that it applies only to the specific string QPR. If $P = P'$, the rule is said to be vacuous.

In general it is neither possible nor desirable to represent a phonology as an explicit list of elementary rules. Usually, in order to preserve the finiteness of the grammar and to express significant generalizations, we must resort to nonelementary rules which, by appropriately economical means, can achieve the effect of a large or even infinite series of elementary rules. It is a widely accepted view that these nonelementary rules should be constructed by means of a schematic notation similar to that developed in the preceding chapters. Accordingly we will allow arrow, slash, and dash to be elementary symbols along with phonological units, thereby implicitly introducing schemata that subsume strings containing these symbols. A nonelementary rule will then be thought of as having the general structure G:X, where G is a symbol designating a particular mode of application and X is a schema subsuming elementary rules, these being referred to as the subrules of G:X. For the time being we will not associate a particular kind of application with a particular kind of schematic expression; for example, we will make no special association between conjunctive ordering and braces. Rather, we will treat all types of schematic expression in a uniform manner, assuming that the rule G:X treats an input string in a way that is determinable entirely from

(i) the input string,
(ii) the mode of application G, and
(iii) the set of elementary rules subsumed under X.

Just what modes of application should be made available to phonological rules is the question to which we now turn.

There are some phonological rules which make changes at no more than one place in any input form. A rule which places a stress on the first vowel of a word or which devoices word-final obstruents is a rule of this type. For any such rule a very simple method of application suffices. Given the string P, we try to find out whether the rule which we wish to apply to P has a subrule whose input is P. If we find such a subrule, we take its output as the output of the application; if we find no such rule we take P itself as the output.

If all phonological rules were of the sort just described, our discussion would be at an end. There are, of course, many rules that can change several different places in an input string; in fact, there may be no principled upper bound to the number of places affected or to the distance separating these places. The Sanskrit rule discussed toward the beginning of Chapter 2, for example, must retroflex every *n* occurring in the appropriate environment, regardless of how many such *n*'s there may be in an input word. An elementary rule, however, changes just one place in an input string. In general, then, we must allow several subrules to be involved in each rule application. One way we might do this is as follows. Instead of stopping after a single one-place application we continue performing such applications until we obtain a string which cannot be further changed. This sort of application, which we shall call iterative, was once proposed by Harms (1966a: 608) and has been discussed by McCawley (1968: 20-22) in connection with sets of rules that are not necessarily elementary in our sense. We can formalize the notion of iterative rule as in (25).

(25) (a) An iterative rule is an expression of the form *I*:X, where X is a schema subsuming elementary rules (the subrules of *I*:X) and *I* is a constant denoting the mode of application next to be defined.

(b) Let P and Q be phonological strings and let *I*:X be an iterative rule. Then *I*:X maps P into Q if and only if there is a sequence (P_1, ..., P_n) of phonological strings such that

(i) $P_1 = P$;

(ii) for each i, $1 \leq i \leq n-1$, some nonvacuous subrule of *I*:X has P_i as input and P_{i+1} as output;

(iii) *I*:X has no nonvacuous subrule with P_n as input; and

(iv) $P_n = Q$.

The sequence (P_1, ..., P_n), if it exists, can be referred to as an application of *I*:X to P.

An iterative version of the Sanskrit nasal retroflexion rule is given in (26).

(26) $$I\text{:}\ \begin{bmatrix} +\text{cor} \\ +\text{nas} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} -\text{ant} \\ (\sigma) \end{bmatrix} \Big/ \ \$^* \begin{bmatrix} +\text{cont} \\ +\text{cor} \\ -\text{ant} \\ -\text{distr} \end{bmatrix} [-\text{cor}]^* \text{—} [+\text{son}]\ \$^*$$

An application of this rule is displayed in (27). Each line other than the first is derived from the preceding line by virtue of the indicated subrule of (26).

(27)	uṣnataraanaam	
	uṣṇataraanaam	(n → ṇ/uṣ—ataraanaam)
	uṣṇataraaṇaam	(n → ṇ/uṣṇataraa—aam)

Note that application (27) is complete despite the fact that the schema in (26) subsumes both the following:

ṇ → ṇ/uṣ—ataraaṇaam
ṇ → ṇ/uṣṇataraa—aam

Because these subrules are vacuous they cannot be applied to the output of (27). If they were allowed to apply, they would do so forever, and (26) would not have provided *uṣṇataraaṇaam* with any realization at all. The desired output could have been obtained only by complicating the expression to the left of the arrow to read

$$\begin{bmatrix} +\text{cor} \\ +\text{nas} \\ +\text{ant} \\ \sigma \end{bmatrix}$$

There is another application of (26) that has the same input and output as (27), namely (28).

(28)	uṣnataraanaam	
	uṣnataraaṇaam	(n → ṇ/uṣnataraa—aam)
	uṣṇataraaṇaam	(n → ṇ/uṣ—ataraaṇaam)

A method of application diametrically opposed to the iterative would be this. Instead of applying subrules in series, changing the string under consideration step by step, we extract just those subrules which have this string as their common input and then make simultaneously all the changes that these subrules call for. This kind of application has been discussed by McCawley (1968: 20-22) and has been proposed by Chomsky and Halle (1968: 392, 398) as the appropriate way of interpreting rules of certain restricted forms.

Formalization of simultaneous rules would be straightforward if every subrule were of the form A → P/R—S, where A is a phonological unit. Then, given the input string $A_1 \ldots A_n$ to which we wish to apply the simultaneous rule N, we would say that the string T was a possible output if T had the form $Q_1 \ldots Q_n$, where for each i one of the following conditions held:

(i) $A_i \rightarrow Q_i/A_1 \ldots A_{i-1} - A_{i+1} \ldots A_n$ is a subrule of N; or
(ii) Q_i is identical to A_i and N has no subrule of the form $A_i \rightarrow Q'_i/A_1 \ldots A_{i-1} - A_{i+1} \ldots A_n$

However, we will also want to account for subrules of the form $P \rightarrow Q/R - S$ where P consists of several phonological units, since certain processes cannot be naturally formulated if such subrules are excluded. For example, metathesis of two adjacent vowels followed by a vowel is naturally formulated as

$$\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \begin{bmatrix}+\text{syl}\\ \tau\end{bmatrix} \rightarrow \tau\sigma/\$^* - [+\text{syl}]\ \$^*$$

but not as

$$\left\{\begin{matrix}[+\text{syl}] \rightarrow \tau/\$^* - \begin{matrix}[+\text{syl}]\\ \tau\end{matrix}\\ [+\text{syl}] \rightarrow \sigma/\$^* \begin{matrix}[+\text{syl}]\\ \sigma\end{matrix} - \end{matrix}\right\} [+\text{syl}]\ \$^*$$

On the other hand, we will make no attempt to accommodate subrules of the form $\varnothing \rightarrow Q/R-S$. It seems that in general we can avoid such subrules with little if any loss of naturalness. Thus a rule that inserts a schwa between the second and third members of a triconsonantal cluster could be written

$$\begin{bmatrix}-\text{syl}\\ \sigma\end{bmatrix} \begin{bmatrix}-\text{syl}\\ \tau\end{bmatrix} \rightarrow \sigma ə \tau/\$^* - [-\text{syl}]\ \$^*$$

just as well as

$$\varnothing \rightarrow ə/\$^*\ [-\text{syl}]\ [-\text{syl}] - [-\text{syl}]\ \$^*$$

In the next chapter we will consider the matter of insertion rules again and find a natural place for them.

In formalizing simultaneous rules we will make use of the notion of overlay, defined as follows. Let $M = P \rightarrow Q/R-S$ and $N = P' \rightarrow Q'/R'-S'$ be elementary rules. Then M overlays N if and only if

(i) $RPS = R'P'S'$,
(ii) R is not longer than R′, and
(iii) S is not longer than S′.

Thus, for example, M will overlay N if

M = uşnataraa → uşnataraa/—naam,
N = n → ṇ/uş—ataraanaam

On the other hand, neither of the following two elementary rules overlays the other:

n → ṇ/uş—astaraanaam
n → ṇ/uşnataraa—aam

We can now define simultaneous rules and their mode of application as in (29).

(29) (a) A simultaneous rule is an expression of the form *S*:X, where X is a schema subsuming elementary rules (the subrules of *S*:X) and *S* is constant designating the mode of rule application defined in (b) below. Each subrule of *S*:X is assumed to have the form P → Q/R—S where $P \neq \varnothing$.

(b) Let P and Q be phonological strings and let *S*:X be a simultaneous rule. Then *S*:X maps P into Q if and only if there is a finite sequence $((P_1, Q_1), \ldots, (P_n, Q_n))$ of ordered pairs of strings such that

(i) n is odd;
(ii) $P = P_1 \ldots P_n$;
(iii) for each even i, $1 < i < n$, the elementary rule
$P_i \rightarrow Q_i/P_1 \ldots P_{i-1} — P_{i+1} \ldots P_n$
is a subrule of *S*:X;
(iv) for each odd i, $1 \leqslant i \leqslant n$, $P_i = Q_i$ and the elementary rule
$P_i \rightarrow Q_i/P_1 \ldots P_{i-1} — P_{i+1} \ldots P_n$
overlays no subrule of *S*:X;
(v) $Q = Q_1 \ldots Q_n$.

The sequence $((P_1, Q_1), \ldots, (P_n, P_n))$ will be called an application of *S*:X with input P and output Q. The elementary rule

$P_i \rightarrow Q_i/P_1 \ldots P_{i-1} — P_{i+1} \ldots P_n$

will be referred to as the ith step of the application (thus the even steps are subrules of *S*:X and the odd steps are vacuous and overlay no subrules of S:X). Frequently we will display a simultaneous application in the form

$$P_1 \begin{matrix} P_2 \\ Q_2 \end{matrix} P_3 \ldots \begin{matrix} P_{n-1} \\ Q_{n-1} \end{matrix} P_n$$

We can create a simultaneous version of the Sanskrit nasal retroflexion rule by simply substituting S for I in (26). We then have (30).

$$(30) \quad S\colon \begin{bmatrix} +\text{cor} \\ +\text{nas} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} -\text{ant} \\ (\sigma) \end{bmatrix} \Big/ \$^* \begin{bmatrix} +\text{cont} \\ +\text{cor} \\ -\text{ant} \\ -\text{distr} \end{bmatrix} [-\text{cor}]^* - [+\text{son}]\$^*$$

This rule gives the same result as (26) but is applied in a radically different way. For example, (30) converts *uṣnataraanaam* into *uṣṇataraaṇaam* by virtue of the following application:

$$\text{uṣ} \begin{matrix} \text{n} \\ \text{ṇ} \end{matrix} \text{ataraa} \begin{matrix} \text{n} \\ \text{ṇ} \end{matrix} \text{aam}$$

This application consists of the following five steps:

1. uṣ → uṣ/—nataraanaam
2. n → ṇ/uṣ—ataraanaam
3. ataraa → ataraa/uṣn—naam
4. n → ṇ/uṣnataraa—aam
5. aam → aam/uṣnataraan—

The even-numbered steps are subrules of (30), and the odd-numbered steps are vacuous elementary rules that overlay no subrules of (30).

Consider the array

$$\text{uṣnataraa} \begin{matrix} \text{n} \\ \text{ṇ} \end{matrix} \text{aam}$$

This sequence consists of three steps:

1. uṣnataraa → uṣnataraa/—naam
2. n → ṇ/uṣnataraa—aam
3. aam → aam/uṣnataraan—

However, we have no application of rule (30) here because the putative first step overlays

n → ṇ/uṣ—ataraanaam.

which is a subrule of (30).

The vowel metathesis rule discussed earlier is a less restricted version of a rule proposed by Chomsky and Halle (1968: 361) for the Kasem language. We give the rule again in (31) in a simultaneous formulation.

(31) $S\colon \begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \begin{bmatrix}+\text{syl}\\ \tau\end{bmatrix} \rightarrow \tau\sigma/\$^{*} - [+\text{syl}]$

This rule changes *piai* into *paii* by virtue of the application

$$\mathrm{p}\ \genfrac{}{}{0pt}{}{\mathrm{ia}}{\mathrm{ai}}\ \mathrm{i}$$

There are three steps:

1. p → p/—iai
2. ia → ai/p—i
3. i → i/pia—

Note that under our formalism (31) is ambiguous. The hypothetical input string *piaia* would be changed into either *paiia* or *piiaa* depending on which of the following applications was taken:

$$\mathrm{p}\ \genfrac{}{}{0pt}{}{\mathrm{ia}}{\mathrm{ai}}\ \mathrm{ia}, \qquad \mathrm{pi}\ \genfrac{}{}{0pt}{}{\mathrm{ai}}{\mathrm{ia}}\ \mathrm{a}$$

In order to get an idea of the relative power of iterative and simultaneous rules we will compare them with certain mapping devices whose formal properties have been well studied. We will assume that a finite sequence of mapping devices is also a mapping device, operating according to principle (32).

(32) Let M = (M_1, ..., M_n) be a sequence of mapping devices (possibly rules), and let I and J be arbitrary strings (not necessarily phonological). Then M maps I into J if and only if there is a sequence I_1, ..., I_{n+1} of strings such that

(i) $I_1 = I$,
(ii) for each i, $1 \leq i \leq n$, M_i maps into I_i into I_{i+1}, and
(iii) $I_{n+1} = J$.

If M maps P into Q, we will sometimes say, using phonological terminology, that M provides Q as a realization of P. Note that a realization must be a phonological string; if M maps P into I alone and I is not a phonological string, then M provides P with no realization. It should be borne in mind that in general a mapping device may provide a given string with zero, one, several, or infinitely many realizations. A device will be called monogenic if it provides each string with exactly one realization. Phonological rules are typically monogenic. Alternative realizations provided to a string by a nonmonogenic rule are said to be in free variation.

One highly generalized and unstructured form of mapping device is the (unrestricted) rewriting system. Chomsky (1963: 357) has discussed devices of this sort from the point of view of string generation, but here we are interested in the way they convert strings into strings. We can assume that each rewriting system is characterized by (i) a finite alphabet of symbols, including the boundary symbol h and the start symbol s, and (ii) a finite set of instructions of the form I → J, where I and J are strings of symbols in the system's alphabet. We define an immediate derivation of a rewriting system M as an expression of the form GIH → GJH, where I → J is an instruction of M, and G and H are strings of symbols in the alphabet of M. A string I is automatically put in the form of hsIh when it is presented as input to a rewriting system, and the system then applies in a series of immediate derivations as long as possible. In other words, rewriting systems operate according to (33).

(33) Let M be a rewriting system and let I and J be strings consisting of symbols of M but containing no occurrences of h or s. Then M maps I into J if and only if there is a sequence (I_1, ..., I_n) strings such that
(i) $I_1 = hsIh$;

(ii) for each i, $1 \leq i \leq n-1$, $I_i \rightarrow I_{i+1}$ is an immediate derivation of M;
(iii) M has no immediate derivation of the form $I_n \rightarrow I'$ for any I'; and
(iv) $I_n = hJh$.

It is a well-established principle that any mapping whatever that can be computed by a finitely statable, well-defined procedure can be effected by a rewriting system (in particular, by a Turing machine, which is a special kind of rewriting system). Hence any theory which allows phonological rules to simulate arbitrary rewriting systems is seriously defective, for it asserts next to nothing about the sorts of mappings these rules can perform. It is rather alarming, then, that we can prove (34).

(34) If M is a monogenic rewriting system there is a finite sequence of phonological rules, each simultaneous or iterative, which provides every phonological string with the same realization that M does.

(*In fact, the behavior of a monogenic rewriting system M can be simulated by a single iterative rule if we suitably code the symbols $D_1, \ldots, D_a$ of that system in terms of phonological units. To do this we can pick two arbitrary distinct phonological units A and B as coding elements. Assuming the symbols $D_1, \ldots, D_a$ to be pairwise distinct and to include all phonological units, we define the coding of D_i, denoted $\langle D_i \rangle$, to be the string $A^i B^{a-i} B$. If $q_1, \ldots, q_n$ are symbols of M (not necessarily distinct), we define $\langle q_1 \ldots q_n \rangle$ (the coding of the string $q_1 \ldots q_n$) to be the string $\langle q_1 \rangle \ldots \langle q_n \rangle$. Thus $\langle GH \rangle = \langle G \rangle \langle H \rangle$ for any strings G and H over the alphabet of M. Furthermore, since the correspondence between the individual symbols and their encodings is one for one, every string is unambiguously recoverable from its encoding.

Now suppose M has the instructions $I_1 \rightarrow J_1, \ldots, I_m \rightarrow J_m$. Since M is monogenic we can assume that each phonological string is the input of exactly one application of M. This follows from the way rewriting systems can be constructed to imitate Turing machines (cf. Chomsky 1963: 358-9). We can therefore assume that $I_i \neq J_i$ for each i. For if any application of M contained an immediate derivation of the form $GI_iH \rightarrow GI_iH$, this application would never terminate, and at least one input string would have no realization, contrary to the assumption that M is monogenic. Consequently if $I_i = J_i$, the instruction $I_i \rightarrow J_i$ would never be invoked in any application and would therefore be eliminable.

Now let Y be the schema $\{\langle D_1\rangle, \ldots, \langle D_a\rangle\}^*$, and let X be the schema $\{\langle I_1\rangle \rightarrow \langle J_1\rangle, \ldots, \langle I_m\rangle \rightarrow \langle J_m\rangle\}$ /Y—Y. We can show that $K \rightarrow L$ is an immediate derivation of M if and only if X subsumes some nonvacuous elementary rule whose input is $\langle K\rangle$ and whose output is $\langle L\rangle$. Suppose first that $K \rightarrow L$ is an immediate derivation of M. Then there will be G, H, and i such that $K = GI_iH$, $L = GJ_iH$, and $I_i \rightarrow J_i$ is an instruction of M. But then X will subsume $\langle I_i\rangle \rightarrow \langle J_i\rangle$ / $\langle G\rangle$ — $\langle H\rangle$, whose input and output are $\langle K\rangle$ and $\langle L\rangle$, respectively. Furthermore, because $I_i \neq J_i$, $\langle I_i\rangle \neq \langle J_i\rangle$. Suppose next that $\langle K\rangle$ and $\langle L\rangle$ are the input and output, respectively, of some elementary rule subsumed under X. Since this elementary rule has the form $\langle I_i\rangle \rightarrow \langle J_i\rangle$ / $\langle G\rangle$ — $\langle H\rangle$ for some G, H, and i ($1 \leqslant i \leqslant m$), we have $\langle K\rangle = \langle GI_iH\rangle$ and $\langle L\rangle = \langle GJ_iH\rangle$. Consequently, because the coding is biunique, $K = GI_iH$ and $L = GJ_iH$. But then, since $I_i \rightarrow J_i$ is an instruction of M, $K \rightarrow L$ is an immediate derivation of M. By considering these remarks and comparing (33) with (25b) one can easily verify that M maps P into Q if and only if *I*:X maps $\langle hsPh\rangle$ into $\langle hQh\rangle$, where P and Q are phonological strings.

Consider again the symbols $D_1, \ldots, D_a$ of M. We can assume that for some k, $3 \leqq k \leqq a$, the symbols $D_k, \ldots, D_a$ are phonological units and that the remaining symbols (which must include at least h and s) are special symbols of M used for internal computation. Thus the encoding of each phonological unit will have the form $A^iB^{a-i}B$ where $i \geqq k$.

We can now design a rule sequence which will literally map P into Q if and only if M does so. The sequence has the form (N_1, ..., N_5) where

$N_1 = S: \{D_k \to \langle D_k \rangle, \ldots, D_a \to \langle D_a \rangle\}/\$^* - \$^*,$
$N_2 = I: \{\varnothing \to \langle hs \rangle /-A^k\$^*,\ \varnothing \to \langle h \rangle/\$^*A^kA^*B^*-\},$
$N_3 = I:\ \varnothing \to \langle hsh \rangle/-,$
$N_4 = I: X,$
$N_5 = S: \{\langle h \rangle \to \varnothing, \langle D_k \rangle \to D_k, \ldots, \langle D_a \rangle \to D_a\}/\$^* - \$^*$

(N_1, N_2) will convert P into ⟨hsPh⟩ when P is nonempty, and N_3 will do the same job when P is empty. N_4 then maps ⟨hsPh⟩ into ⟨hQh⟩ if and only if M maps P into Q, as explained previously. N_5 then converts ⟨hQh⟩ into Q.*)

It would seem, then, that a rule formalism which permits both simultaneous and iterative rules is excessively powerful. Notice that we reduce this power very little if we exclude simultaneous rules and permit only iterative ones. (*For let M, N_1, ..., N_5 be as above. Obtain N'_1 from N_1 by replacing *S* with *I* and deleting $D_k \to \langle D_k \rangle$ and $D_{k+1} \to \langle D_{k+1} \rangle$ from the braced expression. Obtain N'_5 from N_5 by simply replacing *S* with *I*. Then (N'_1, N_2, N_3, N_4, N'_5) will provide each phonological string that is free of occurrences of D_{k+1} and D_k with the same realization that M does. Strings containing occurrences of D_k and D_{k+1} will not in general be handled correctly, of course; D_k and D_{k+1} have been sacrified to the cause of coding.*)

Suppose we try the polar alternative of excluding iterative rules altogether and allowing simultaneous rules only. We will try to show that simultaneous rules are equivalent to finite-state machines, which are highly restricted in structure and are incapable, in fact, of performing a vast array of computable mappings, though the mappings they do perform seem to include most of those that arise in phonology.

In its most general form a finite-state machine is characterized by

(i) a finite alphabet of input symbols,
(ii) a finite alphabet of output symbols,
(iii) a finite set of symbols referred to as states,
(iv) a subset of states designated as left terminal,
(v) a subset of states designated as right terminal, and
(vi) a finite set of instructions of the form $q\ K \to L\ q'$,

where q and q′ are states, K is an input symbol, and L is a string of output symbols.

By a computation of a finite-state machine M we will mean an expression of the form $q_1K_1 \rightarrow L_1 \ldots q_nK_n \rightarrow L_nq_{n+1}$ where for each i, $1 \leq i \leq n$, $q_iK_i \rightarrow L_iq_{i+1}$ is an instruction of M. $K_1 \ldots K_n$ will be referred to as the input of the computation, and $L_1 \ldots L_n$ will be referred to as the output. If q_1 is a left terminal state of M and q_{n+1} is a right terminal state of M, the computation is said to be terminated. By definition M maps I into J if and only if some terminated computation of M has I as input and J as output.

A simple example of a finite-state machine is the one characterized as follows:

Input alphabet: A, B
Output alphabet: A, B, C
States: 0, 1, 2
Left terminal states: 0
Right terminal states: 0, 1
Instructions: 0A → A1
0A → C2
0B → B0
1A → C0
1B → B1
2A → A1

One of the computations of this machine is

0B → B0A → A1B → B1A → C0B → B0A → C2A → A1

The input of the computation is BABABAA, and the output is BABCBCA. Since the first symbol of the computation is a left-terminal state of the machine and the last symbol is a right terminal state, the computation is terminated. Hence we may conclude that the machine maps BABABAA into BABCBCA.

We will say that a finite-state machine is right-deterministic if it has exactly one left terminal state and, for each state q and each input symbol K, exactly one instruction of the form qK → Lq′.

If a finite-state machine has exactly one right terminal state and, for each state q and input symbol K, exactly one instruction of the form q′K → Lq, we will say that the machine is left-deterministic. We denote by lt(M) the unique left terminal state of a right-deterministic machine M, and use rt(M) for the unique right terminal state of a left-deterministic machine M.

We will be especially concerned with two varieties of finite state machines called right transducers and left transducers. A right transducer is a right-deterministic finite-state machine all of whose states are right terminal. Clearly a device M of this sort will convert I into J if and only if J is the output of that unique computation which has I as input and had lt(M) as its leftmost state. A left transducer is defined in completely symmetric manner as a left-deterministic finite state machine all of whose states are left terminal. A right transducer is also known as a generalized sequential machine (Ginsburg 1962: 5, 20), the unique left terminal state being also known as the start state or the initial state.

Although we have been regarding mapping devices only as mechanisms for converting strings into strings, this being the function of phonological rules, we can also look upon them as devices for defining or representing sets of strings. Suppose we say that a phonological string P is accepted by a mapping device M if and only if M provides P with at least one realization. Those phonological strings that are accepted by the device constitute the defined set. It is well known that a set is regular if and only if it is defined by some finite-state machine. Hence phonological schemata and finite-state machines are entirely equivalent in their capacity to represent sets.

If we wish to construct a finite-state machine solely for the purpose of accepting strings there is no point in providing it with output symbols. We might just as well let each instruction be of the form qK → q′. Then the machine accepts P if and only if it maps P into ∅. A finite-state machine of this sort is called a finite-state automaton. If such a machine is right- (left-) deterministic it is said to be a right- (left-) automaton.

What we wish to show now is (35).

(35) If N is a monogenic simultaneous rule then there is a left transducer M_1 and a right transducer M_2 such that (M_1, M_2) provides each phonological string with the same realization that N does.

(*We let N = *S*:X be an arbitrary monogenic simultaneous rule, fixed throughout the proof. If N has no subrules it leaves input strings unchanged, and there is a trivial transducer that can accomplish this identity mapping. Henceforth we assume N to have at least one subrule.

Without loss of generality we can assume that $X = \{X_1, \ldots, X_m\}$ is a primitive schema. Each of the X_i will then have the form $Y_i \rightarrow Y'_i/U_i — V_i$, where Y_i, Y'_i, U_i, and V_i subsume only phonological strings. To see this consider each of the X_i in turn, letting $(Z_1, E_1, \ldots, Z_p, E_p, Z_{p+1})$ be the longest sequence of schemata such that $X_i = Z_1E_1 \ldots Z_pE_pZ_{p+1}$ and such that each E_h $(h = 1, \ldots, p)$ is an arrow, a slash, or a dash. Since this sequence is maximally long none of the Z_j can have the form $Z'E'Z''$ where E' is arrow, slash, or dash and Z' and Z'' are schemata. Hence if arrow, slash, or dash appears anywhere within a Z_j it must be inside a braced or starred expression (these being the only nonelementary schematic expressions allowed in primitive schemata). Now, any braced expression that is a proper part of a primitive schema must be enclosed in starred parentheses. Consequently, an arrow, slash, or dash appearing within any of the Z_j must be inside a starred expression. However, if an arrow, slash or dash appeared inside a starred expression it could be repeated an unbounded number of times in subsumed strings, and X would subsume some expressions that were not elementary rules.

Because simultaneous rules have no subrules of the form $\varnothing \rightarrow Q/R — T$, we can assume that the Y_i subsume only nonempty strings. We can also assume that each Y'_i is a phonological string, since if *S*:X is indeed monogenic none of the Y'_i need subsume more than one phonological string. We will set $S_i = Y'_i$ and $W_i = Y_iV_i$.

The first step in creating a pair of transducers to simulate N is to construct, for each $k = 1, \ldots, m$, two left automata $\overline{W}_k$ and $\overline{V}_k$ that represent the same regular sets as W_k and V_k, respectively, and two right automata $\bar{U}_k$ and $\bar{Y}_k$ that represent the same regular sets as U_k and Y_k, respectively. In addition we define $\bar{U}_0 = \bar{Y}_0 = \overline{W}_0 = \overline{V}_0$ to be the automaton having just the one state 0 and having the instruction $0A \rightarrow 0$ for each phonological unit A. 0 is taken to be both left terminal and right terminal. This device is at once a right automaton and a left automaton, and it accepts every phonological string. Suppose now we define $((z_0, \ldots, z_m))_k = z_k$ for any $(m+1)$ -tuple $(z_0, \ldots, z_m)$ and any $k = 0, \ldots, m$. An $(m+1)$ -tuple x will be said to be a W-state (resp.: V-state, U-state) if and only if $(x)_k$ is a state of $\overline{W}_k$ (resp.: $\overline{V}_k$, $\bar{U}_k$) for each $k = 0, \ldots, m$. Henceforth we use the letters w, v, and u to stand for W-states, V-states, and U-states respectively. The letter y will range over states of all the $\bar{Y}_i$. Also we define

$$\bar{\bar{w}} = (rt(\overline{W}_0), \ldots, rt(\overline{W}_m))$$
$$\bar{\bar{v}} = (rt(\overline{V}_0), \ldots, rt(\overline{V}_m))$$
$$\bar{\bar{u}} = (lt(\bar{U}_0), \ldots, lt(\bar{U}_m))$$

We now construct the first transducer M_1. The input alphabet of M_1 is the phonological alphabet and the output alphabet consists of triples of the form (w, v, A). The states of M_1 are couples of the form (w, v). Each state of M_1 is left terminal, and M_1 has the unique right terminal state $(\bar{\bar{w}}, \bar{\bar{v}})$. For each input symbol A and each state (w, v), M_1 has the instruction $(w', v')A \rightarrow (w', v', A)(w, v)$ where for each $k = 0, \ldots, m$, $(w')_kA \rightarrow (w)_k$ and $(v')_kA \rightarrow (v)_k$ are instructions of $\overline{W}_k$ and $\overline{V}_k$, respectively. M_1 is a left transducer which, when presented with an input string $A_1 \ldots A_n$, responds with the output string $(w_1, v_1, A_1) \ldots (w_n, v_n, A_n)$ where the conditions of (35) hold.

(35) (a) for each k, $0 \leq k \leq m$, $(w_n)_kA_n \rightarrow rt(\overline{W}_k)$ and $(v_n)_kA_n \rightarrow rt(\overline{V}_k)$ are instructions of $\overline{W}_k$ and $\overline{V}_k$, respectively.

(b) for each k, $0 \leq k \leq m$, and each i, $1 \leq i \leq n-1$, $(w_i)_kA_i \rightarrow (w_{i+1})_k$ and $(v_i)_kA_i \rightarrow (v_{i+1})_k$ are instructions of $\overline{W}_k$ and $\overline{V}_k$, respectively.

The second transducer M_2 is constructed as follows. The input alphabet of M_2 is identical to the output alphabet of M_1, and the output alphabet of M_2 is again the phonological alphabet. The states of M_2 are triples of the form (u, y, k) where $0 \leq k \leq m$, y is a state of $\bar{Y}_k$, and u is a U-state. M_2 has the unique left terminal state $(\bar{u}, 0, 0)$, and all of the states of M_2 are right terminal. The instructions of M_2 are given by (36).

(36) For each input symbol (w, v, A) and each state (u, y, k), M_2 has the instruction $(u, y, k)\ (w, v, A) \rightarrow Q(u', y', k')$ where
- (a) for each k, $0 \leq k \leq m$, $(u)_k A \rightarrow (u')_k$ is an instruction of $\bar{U}_k$;
- (b_1) if y is a right terminal state of $\bar{Y}_k$ and $(v)_k$ is a left terminal state of $\bar{V}_k$, then
  - (i) k′ is the highest integer such that $(u)_{k'}$ is a right terminal state of $\bar{U}_{k'}$ and $(w)_{k'}$ is a left terminal state of $\bar{W}_{k'}$ (such an integer will always exist because 0, at least, is such an integer),
  - (ii) $Q = A$ or $S_{k'}$ according as k′ is equal to or greater than zero, and
  - (iii) $1t(\bar{Y}_{k'})A \rightarrow y'$ is an instruction of $\bar{Y}_{k'}$;
- (b_2) if y is not a right terminal state of $\bar{Y}_k$ or $(v)_k$ is not a left terminal state of $\bar{V}_k$, then
  - (i) $k' = k$,
  - (ii) $Q = \varnothing$, and
  - (iii) $yA \rightarrow y'$ is an instruction of $\bar{Y}_k = \bar{Y}_{k'}$.

Each string $(w_1, v_1, A_1) \ldots (w_n, v_n, A_n)$ produced as output by M_1 will be the input of exactly one terminated computation of the right transducer M_2, and this computation will have the general form $K_0 \ldots K_n$ where

$$K_0 = (u_0, y_0, k_0) \text{ and}$$
$$K_i = (w_i, v_i, A_i) \rightarrow T_i(u_i, y_i, k_i) \text{ for } i = 1, \ldots, n.$$

(M_1, M_2), then, maps $A_1 \ldots A_n$ into $T_1 \ldots T_n$. We need to show that the simultaneous rule N also maps $A_1 \ldots A_n$ into $T_1 \ldots T_n$.

Since (u_0, y_0, k_0) is the left-terminal state of M_2 we have (37).

(37) $u_0 = \bar{\bar{u}}$, $y_0 = 0$, and $k_0 = 0$.

For each $i = 1, \ldots, n$ and each $r = 0, \ldots, m$, the expression $(u_{i-1})_r A_i \rightarrow (u_i)_r$ is an instruction of $\bar{U}_r$. This follows by virtue of (36a). Because of (37) $(u_0)_r = \mathrm{lt}(\bar{U}_r)$. Also $\bar{U}_r$ is right-deterministic. Hence

$$(u_0)_r A_1 \rightarrow (u_1)_r \ldots A_i \rightarrow (u_i)_r$$

is that unique computation of $\bar{U}_r$ that begins with $\mathrm{lt}(\bar{U}_r)$ and has $A_1 \ldots A_i$ as input. Therefore $A_1 \ldots A_i$ is accepted by $\bar{U}_r$ if and only if $(u_i)_r$ is a right-terminal state of $\bar{U}_r$. Reasoning in a similar way from (35) we can show that $A_i \ldots A_n$ is accepted by $\bar{W}_r$ (resp.: $\bar{V}_r$) if and only if $(w_i)_r$ (resp.: $(v_i)_r$) is a left-terminal state of $\bar{W}_r$ (resp.: $\bar{V}_r$). Hence we have (38).

(38) If $1 \leq i \leq n$ and $1 \leq r \leq m$, then
 (a) $A_1 \ldots A_i$ is subsumed under U_r if and only if $(u_i)_r$ is a right-terminal state of $\bar{U}_r$; and
 (b) $A_i \ldots A_n$ is subsumed under W_r (resp.: V_r) if and only if $(w_i)_r$ (resp.: $(v_i)_r$) is a left-terminal state of $\bar{W}_r$ (resp.: $\bar{V}_r$).

π For each $i = 1, \ldots, n$ we will say that i is a type 1 or type 2 index according as the instruction

$$(u_{i-1}, y_{i-1}, k_{i-1})\,(w_i, v_i, A_i) \rightarrow T_i(u_i, y_i, k_i)$$

satisfies condition $(36b_1)$ or $(36b_2)$. Note that these conditions are mutually exclusive and exhaust the possibilities, so that each i from 1 through n must be either a type 1 index or a type 2 index, but cannot be both.

Now suppose that i is a type 1 index such that $k_i = 0$. By $(36b_1.ii)$ $T_i = A_i$. Furthermore, if $i < n$, then $i + 1$ is a type 1 index; this follows from the fact that $y_i = 0 = (v_{i+1})_0 = (v_{i+1})_{k_i}$ is both a right-terminal state and a left-terminal state of $\bar{Y}_{k_i} = \bar{Y}_0 = \bar{V}_0 = \bar{V}_{k_i}$ (cf. $36b_1$). By $(36b_1.i)$ there is no $r > k_i$ such that $(u_{i-1})_{k_i}$ is a right-terminal state of $\bar{U}_r$ and $(w_i)_{k_i}$ is a left-terminal state of $\overline{W}_r$. Hence by (38) there is no $r > k_i$ such that $A_1 \ldots A_{i-1}$ is subsumed under U_r and $A_i \ldots A_n$ is subsumed under W_r. Consequently, since $W_r = Y_r V_r$ for each $r = 1, \ldots, m$, there is no j such that $A_i \ldots A_j$ is subsumed under Y_r and $A_{j+1} \ldots A_n$ is subsumed under V_r (taking $A_{j+1} \ldots A_n = \varnothing$ if $j = n$). Therefore there is no j such that

$$A_i \ldots A_j \rightarrow T_i \ldots T_j / A_1 \ldots A_{i-1} - A_{j+1} \ldots A_n$$

is a subrule of N.

We can immediately extend the results of the preceding paragraph to (39).

(39) Let i and j be indices such that $i \leqq j$ and such that each i', $i \leqq i' \leqq j$, is a type 1 index for which $k_{i'} = 0$. Then
(a) $T_i \ldots T_j = A_i \ldots A_j$;
(b) if $j < n$, $j + 1$ is a type 1 index;
(c) there are no i', i'' such that $i \leqq i' \leqq i'' \leqq j$ and such that the elementary rule
$A_{i'} \ldots A_{i''} \rightarrow T_{i'} \ldots T_{i''} / A_1 \ldots A_{i'-1} - A_{i''+1} \ldots A_n$
is a subrule of N. Consequently the elementary rule
$A_i \ldots A_j \rightarrow T_i \ldots T_j / A_1 \ldots A_{i-1} - A_{i+1} \ldots A_n$
does not overlay any subrule of N.

Now let i be a type 1 index for which $k_i > 0$. Because of (38) $A_1 \ldots A_{i-1}$ is subsumed under U_{k_i} and $A_i \ldots A_n$ is subsumed under W_{k_i}. Also, $T_i = S_{k_i}$. Now let j be the highest integer not greater than n such that each h, $i < h \leqq j$, is a type 2 index. (If $j = i$, then of course there is no such h; but if $j > i$, then j at least will be such an h.) For each such h we have $T_h = \varnothing$ by ($36b_2$.ii), and therefore $T_i = S_{k_i} = T_i \ldots T_j$. Now because of ($36b_2$), we have (i) and (ii):

(i) $k_h = k_h'$ for $i \leqq h \leqq h' \leqq j$, and
(ii) $y_{h-1} A_h \rightarrow y_h$ is an instruction of $\overline{Y}_{k_i}$ for each h, $i < h \leqq j$.

Also, because of ($36b_1$.iii), $1t(\overline{Y}_{k_i}) A_i \rightarrow y_i$ is an instruction of

$\bar{Y}_{k_i}$. Recall, too, that $\bar{Y}_{k_i}$ is right-deterministic. Consequently, for $i \leq h \leq j$,

$$1t(\bar{Y}_{k_i})\ A_i \rightarrow y_i \ldots A_h \rightarrow y_h$$

is that unique computation of $\bar{Y}_{k_i}$ which has $A_i \ldots A_h$ as input and $1t(\bar{Y}_{k_i})$ as its leftmost state. Therefore $A_i \ldots A_h$ is accepted by $\bar{Y}_{k_i}$ if and only if y_h is a right-terminal state of $\bar{Y}_{k_i}$. Therefore $A_i \ldots A_h$ is subsumed under Y_{k_i} if and only if y_h is a right-terminal state of $\bar{Y}_{k_i}$. Now consider the two cases $j < n$ and $j = n$.

Case 1: $j < n$. $j+1$ must then be type 1 index. But then because of ($36b_1$) y_j is a right-terminal state of $\bar{Y}_{k_i}$ ($= \bar{Y}_{kj}$) and $(v_{j+1})_{k_j}$ is a left-terminal state of $\bar{V}_{k_j}$ ($= \bar{V}_{k_i}$). From this it follows that $A_i \ldots A_j$ is subsumed under Y_{k_i} and $A_{j+1} \ldots A_n$ is subsumed under V_{k_i}.

Case 2: $j = n$. Because $A_i \ldots A_n$ is subsumed under W_{k_i} and $W_{k_i} = Y_{k_i}V_{k_i}$ one of the following conditions must hold:
(i) $\varnothing$ is subsumed under Y_{k_i} and $A_i \ldots A_n$ is subsumed under V_{k_i};
(ii) for some h, $i \leq h < j$, $A_i \ldots A_h$ is subsumed under Y_{k_i} and $A_{h+1} \ldots A_n$ is subsumed under V_{k_i}; or
(iii) $A_i \ldots A_n$ is subsumed under Y_{k_i} and $\varnothing$ is subsumed under V_{k_i}.

(i) is not satisfied because Y_{k_i} subsumes only nonempty strings (recall that rule N has no insertion subrules). (ii) would imply that y_h is a right-terminal state of $\bar{Y}_{k_i}$ and $(v_{h+1})_{k_i}$ is a left-terminal state of $\bar{V}_{k_i}$; therefore, since $k_h = k_i$, $h+1$ would be a type 1 index (cf. $36b_1$). This would contradict the assumption that each integer greater than i and equal to or less than j is a type 2 index. The only remaining possibility is (iii).

What we have managed to show is (40).

(40) Let i be a type 1 index for which $k_i > 0$ and let j be the highest integer such that each h, $i < h \leq j$, is a type 2 index. Then

> $A_i \ldots A_j \rightarrow T_i \ldots T_j / A_1 \ldots A_{i-1} - A_{j+1} \ldots A_n$
> is subsumed under $Y_{k_i} \rightarrow S_{k_i}/U_{k_i} - V_{k_i}$ and is therefore a subrule of N.

Now let $s_1, \ldots, s_p$ be all and only the type 1 indices for which $k_{s_i} > 0$. $(1 \leqq i \leqq p)$. We can assume $s_i < s_{i'}$ if $i < i'$. For each i, $1 \leqq i \leqq p$, let t_i be the highest index such that each h, $s_i < h \leqq t_i$, is a type 2 index. Also, define $t_0 = 0$ and $s_{p+1} = n+1$. Clearly, $t_i < s_{i+1}$. Now for $i = 0, \ldots, p$ we define

$$\begin{aligned} P_{2i+1} &= A_{t_i} \ldots A_{s_{i+1}-1}; \\ Q_{2i+1} &= T_{t_i} \ldots A_{s_{i+1}-1}; \\ P_{2i+2} &= A_{s_i} \ldots A_{t_i}; \text{ and} \\ Q_{2i+2} &= T_{s_i} \ldots T_{t_i}. \end{aligned}$$

From these definitions it follows that $A_1 \ldots A_n = P_1 \ldots P_{2p+1}$ and $T_1 \ldots T_n = Q_1 \ldots Q_{2p+1}$. Now for each j, $1 \leqq j \leqq 2p+1$, let M_j be the elementary rule

$$P_j \rightarrow Q_j / P_1 \ldots P_{j-1} - P_{j+1} \ldots P_{2p+1}$$

If j is even, then $j = 2i+2$ for some i, $0 \leqq i \leqq p$. Then M_j is the elementary rule

$$A_{s_i} \ldots A_{t_i} \rightarrow T_{s_i} \ldots T_{t_i} / A_1 \ldots A_{s_i - 1} - A_{t_i + 1} \ldots A_n$$

which, by virtue of (40), is a subrule of N. If j is odd then $j = 2i+1$ for some i, $0 \leqq i \leqq p$, and then M_j is the elementary rule

$$A_{t_i+1} \ldots A_{s_{i+1}-1} \rightarrow T_{t_i+1} \ldots T_{i+1} / A_1 \ldots A_{t_i} - A_{s_{i+1}} \ldots A_n$$

There are now two cases to consider, $t_i = s_{i+1}$ and $t_i < s_{i+1} - 1$

Case 1: $t_i = s_{i+1} - 1$. Then $P_j = Q_j = \varnothing$ and M_j overlays no subrule of N. The reason for this is that M_j is an insertion rule, and an insertion can overlay only another insertion rule. However, N has no insertion subrules.

Case 2: $t_i < s_{i+1} - 1$. We consider first the type of index that $t_i + 1$ is.

Subcase 1. $i = 0$. Then $t_i + 1 = 1$, which is a type 1 index because $k_0 = 0$, $y_0 = 0$ is a right-terminal state of $\bar{Y}_0$, and $(v_1)_0 = 0$ is a left-terminal state of $\bar{V}_0$.

Subcase 2. $i > 0$. Then $t_i + 1$ is less than $n = s_{p+1}-1$ and is a type 1 index. For if $t_i + 1$ were a type 2 index t_i would not be the highest index such that each h, $s_i < h \leq t_i$, is a type 2 index, contrary to the definition of t_i.

Now although $t_i + 1$ is a type 1 index it is not among the $s_1, \ldots, s_p$, because it is larger than s_i and less than s_{i+1}. Therefore $k_{t_i+1} = 0$. Hence, because of (39c), M_j overlays no subrule of N. Furthermore, (39a) implies that $P_j = Q_j$.

We have now shown that when j is even M_j is a subrule of N and that when j is odd $P_j = Q_j$ and M_j overlays no subrule of N. Consequently the sequence $((P_1, Q_1), \ldots, (P_{2p+1}, Q_{2p+1}))$, whose jth step is M_j, is an application of N. But this application has $P_1 \ldots P_{2p+1} = A_1 \ldots A_n$ as input and $Q_1 \ldots Q_{2p+1} = T_1 \ldots T_n$ as output. Thus N maps $A_1 \ldots A_n$ into $T_1 \ldots T_n$. Q.E.D.*)

A left (right) transducer takes into account only what is to the right (left) of an input symbol in determining how to replace that symbol in the output. Schützenberger (1961) has investigated a finite-state device which takes into account both the left and right contexts of each input symbol. This machine, referred to simply as a finite transducer, is characterized by the following structures:

(i) Two finite alphabets $\bar{I}$ and $\bar{O}$ (the input symbols and the output symbols).

(ii) A right automaton $\bar{U}$ whose input alphabet is $\bar{I}$ and whose states are all right terminal.

(iii) A left automaton $\bar{V}$ whose input alphabet is $\bar{I}$ and whose states are all left terminal.

(iv) A finite set of output instructions of the form $uKv \rightarrow L$, where u is a state of $\bar{U}$, K is an input symbol, v is a state of $\bar{V}$, and L is a string of output symbols. There is exactly one instruction $uKv \rightarrow L$ for each uKv.

The way a finite transducer works is this. Suppose $K_1, \ldots, K_n$ are input symbols. For each i $(i = 1, \ldots, n)$ let $\left\{\begin{matrix} u_i \\ v_i \end{matrix}\right\}$ be the $\left\{\begin{matrix} \text{rightmost} \\ \text{leftmost} \end{matrix}\right\}$ state in that terminated computation of $\left\{\begin{matrix} \bar{U} \\ \bar{V} \end{matrix}\right\}$ which has $\left\{\begin{matrix} K_i \ldots K_{i-1} \\ K_{i+1} \ldots K_n \end{matrix}\right\}$ as its input, and suppose $u_i K_i v_i \rightarrow L_i$ is the output instruction for $u_i K_i v_i$. Then the transducer converts $K_1 \ldots K_n$ into $L_1 \ldots L_n$.

Schützenberger has pointed out that an ordered pair of machines, one of which is a left transducer and the other of which is a right transducer, is equivalent to some finite transducer in the mapping its effects. Consequently every monogenic simultaneous rule is similarly equivalent to some finite transducer. Thus a theory which permits only simultaneous rules in a phonological description embodies a very strong hypothesis concerning the sorts of string mappings that are possible in phonology, for finite transducers rank next to the bottom in the following hierarchy.

1. Rewriting machines (in particular, Turing machines).
2. Linear bounded transducers.
3. Pushdown-store transducers.
4. Finite transducers.
5. Right transducers; left transducers.

It is well known that every mapping effected by a device of type n in this list can be effected by a device of type $n-1$ $(n = 2, \ldots, 5)$, but not conversely. (Right transducers and left transducers are not comparable with each other in this way because some right transducers can do things that left transducers can't and vice-versa.) There seem, in fact, to be few phonological processes that exceed the capacity of finite transducers; the ones known to me belong to the very restricted types to be discussed in Chapter 7. In at least one respect, then, simultaneous rules are far more appropriate to phonology than iterative ones.

If the finite-state claim is correct, there is a clear choice between two recent proposals concerning the way such processes as contraction, metathesis, gemination, and degemination should be handled. Postal (1969: 298) has suggested the use of variables ranging over lists of feature specifications. He had in mind a vowel-doubling rule, but his considerations extend to the other types of processes just mentioned. Thus according to Postal's proposal the rule mentioned earlier that metathesized two vowels before another vowel could be written:

$$\begin{bmatrix}+\text{syl}\\ \text{A}\end{bmatrix}\begin{bmatrix}+\text{syl}\\ \text{B}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{syl}\\ \text{B}\end{bmatrix}\begin{bmatrix}+\text{syl}\\ \text{A}\end{bmatrix} / - [+\text{syl}]$$

Under our view of brackets as representing set intersection, Postal's variable can in most cases be regarded as a unit variable. It is possible, therefore, to simplify slightly the above rule to

$$\begin{bmatrix}+\text{syl}\\ \text{A}\end{bmatrix}\begin{bmatrix}+\text{syl}\\ \text{B}\end{bmatrix} \rightarrow \text{B A} / - [+\text{syl}]$$

This, of course, is just a notational variant of our rule (31).

Another way of writing the kind of rule under discussion is with transformational format:

$$\begin{array}{ccccc} \text{X}, & [+\text{syl}], & [+\text{syl}], & [+\text{syl}], & \text{Y} \\ 1 & 2 & 3 & 4 & 5 \end{array} \Rightarrow 1, 3, 2, 4, 5$$

Langacker (1969: 858-9) has correctly observed that all phonological rules can be written in this format, and has proposed that this be done. Notice, however, that in this format integers are used as general string variables. Consequently many nonfinite-state operations can be performed, such as the reduplication of whole words. Such reduplications occur, to be sure (for example, in Indonesian). However, reduplication seems to be a morphological process spelling out grammatical elements or features (e.g. plurality or distribution) and seems to have no phonological origin or motivation. It belongs rather with what Chomsky and Halle (1968: 9-11) have referred to as readjustment rules. I would claim that phonological rules proper never require general string variables for their expression and would therefore reject Langacker's proposal as insufficiently restrictive.

5

LINEAR RULES

The simultaneous rule model is, of course, just one of a number of conceivable formalisms that would impose a finite-state restriction on phonological rules. One could, for example, propose that phonological rules be finite transducers in the literal sense. No one would take such a suggestion seriously because of the linguistic inappropriateness of the formulations it would require. We will, however, give serious consideration to another type of rule, so far employed very rarely in generative phonology, which shares some of the properties of the iterative type but which, like the simultaneous rule, is a finite-state device in its mapping power. This new type of rule, to be called linear, has two subvarieties which we will refer to as the right-linear and the left-linear.

We consider first right-linear rules. To describe how these function it will be convenient to use the notion of right-overlap. If $M = P \rightarrow Q/R - S$ and $N = P' \rightarrow Q' \rightarrow Q'/R' - S'$ are elementary rules, we will say that M right-overlaps N if and only if

(i) $RPS = R'P'S'$,
(ii) R is not longer than R′, and
(iii) RP is longer than R′.

Thus M right-overlaps N if

M = uṣnataraa → uṣnataraa/—naam,
N = n → ṇ/uṣ—ataraanaam

or if

M = nata → nata/—pikkappɨkai
N = a → á/nat—pikkappɨkai

However, M does not right-overlap N if

$$\begin{aligned} M &= \text{uṣ} \rightarrow \text{uṣ}/\text{—nataraanaam},\\ N &= \text{n} \rightarrow \text{ṇ}/\text{uṣ—ataraanaam}\end{aligned}$$

or if

$$\begin{aligned} M &= \text{watunis} \rightarrow \text{watunis}/\text{—ˀas},\\ N &= \emptyset \rightarrow \text{a}/\text{watunis—ˀas}\end{aligned}$$

We now formalize right-linear rules as in (41).

(41) (a) A right-linear rule is an expression of the form *R*:X where X is a schema subsuming elementary rules subrules of *R*:X) and *R* is a constant denoting the mode of application next to be defined. There is no restriction on the form of subrules.

(b) Let P and Q be phonological strings and let *R*:X be a right-linear rule. Then *R*:X maps P into Q if and only if there is a sequence $((P_1, Q_1), \ldots, (P_n, Q_n))$ of ordered pairs of strings such that

(i) n is odd;

(ii) $P = P_1 \ldots P_n$;

(iii) for each even i, $2 \leq i \leq n-1$, the elementary rule

$$P_i \rightarrow Q_i/Q_1 \ldots Q_{i-1} - P_{i+1} \ldots P_n$$

is a subrule of *R*:X;

(iv) for each odd i, $1 \leq i \leq n$, $P_i = Q_i$ and the elementary rule

$$P_i \rightarrow Q_i/Q_1 \ldots Q_{i-1} - P_{i+1} \ldots P_n$$

does not right-overlap any subrule of *R*:X;

(v) $Q = Q_1 \ldots Q_n$.

The sequence $((P_1, Q_1), \ldots, (P_n, Q_n))$, which we will write also as

$$P_1 \; \frac{P_2}{Q_2} \; P_3 \ldots \frac{P_{n-1}}{Q_{n-1}} \; P_n,$$

will be referred to as an application of *R*:X with input P and output Q. The ith rightward step of the application is defined as the elementary rule $P_i \rightarrow Q_i/Q_1 \ldots Q_{i-1}$

— $P_{i+1}\ldots P_n$. Thus the even-numbered steps are subrules of *R*:X and the odd-numbered steps right-overlap no subrules of *R*:X.

The Sanskrit nasal retroflexion rule can be used again to illustrate the linear modes of application. Using the same schema as we did in the simultaneous rule (30), we can construct the right-linear rule (42).

$$(42)\quad R\colon \begin{bmatrix}+\text{nas}\\ +\text{cor}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}-\text{ant}\\ (\sigma)\end{bmatrix} / \$^{*} \begin{bmatrix}+\text{cor}\\ +\text{cont}\\ -\text{ant}\\ -\text{distr}\end{bmatrix} [-\text{cor}]^{*} \text{—} [+\text{son}]\$^{*}$$

This will give the same results as the simultaneous rule. In particular, it sill convert *uṣnataraanaam* into *uṣṇataraaṇaam* by virtue of the same application, namely:

$$\text{uṣ}\ \overset{\text{n}}{\underset{\text{ṇ}}{}}\ \text{ataraa}\ \overset{\text{n}}{\underset{\text{ṇ}}{}}\ \text{aam}$$

Here, however, the application is regarded as consisting of the following rightward steps:

1. uṣ → uṣ/—nataraanaam
2. n → ṇ/uṣ—ataraanaam
3. ataraa → ataraa/uṣṇ—naam
4. n → ṇ/uṣṇataraa—aam
5. aam → aam/uṣṇataraaṇ—

Notice that the dash appears in the leftmost possible position in the first step and occupies positions successively farther to the right in subsequent steps. Furthermore the output of each step other than the last is the input to the next step. Consider in particular the even-numbered steps, which are also subrules of (42). Since the other steps are vacuous, the input to the application is also the input to the second step, the output of the second step is the input to the fourth step, and the output of the fourth step is the output of the application as a whole. Now observe that 2 and 4 are precisely those subrules which were invoked in application (27) of the iterative version of the rule (26). Thus the right linear rule, though formally very similar to the simultaneous rule, is in another respect quite like the iterative rule. In general, the function of *R* is to restrict iterative applications to those which proceed through the input string in a strictly left-to-right manner, stopping when the right end of the string is reached even if the string in its current form is the input to a further subrule. These conditions on application are sufficient to reduce the power of iterative rules to that of finite-state devices. For it is possible to demonstrate (43).

(43) If N is a monogenic right-linear rule there is a left-transducer M_1 and a right transducer M_2 such that (M_1, M_2) provides each nonempty phonological string with exactly the same realization that N does.

(* We can let N = *R*:X. As in the proof of (35) we can assume that $X = \{X_1, \ldots, X_m\}$ is a primitive schema in which each X_i has the form $Y_i \rightarrow S_i/U_i - V_i$, where S_i is a phonological string and Y_i, U_i, and V_i subsume only phonological strings.

Suppose first that N has no subrules of the form $\emptyset \rightarrow P/Q - R$. The construction of (M_1, M_2) proceeds exactly as the proof of (35) down through the construction of M_1. M_2 has the states, input alphabet, and output alphabet described in the proof of (35), and a similar designation of left and right terminal states, but a different set of instructions. Specifically, M_2 will have, for each state (u, y, k) and each input symbol (w, v, A), the instruction $(u, y, k)(w, v, A) \rightarrow Q(u', y', k')$ where all the conditions of (44) hold.

(44) (a) For each r, $1 \leq r \leq m$, $(u')_r$ is the rightmost state of that computation of $\bar{U}_r$ which has $(u)_r$ as its leftmost state and Q as its input. (Notice that u′ must be identical to u if $Q = \emptyset$.)

(b_1) (Identical to ($36b_1$).)

(b_2) (Identical to ($36b_2$).)

The proof that (M_1, M_2) does the same work as N will not be given, since it follows the same general lines as the corresponding portion of the proof of (35). The minor differences that exist between the two proofs stem from the discrepancy between (44a) and (36a).

Suppose now that N has some subrules of the form $0 \rightarrow P/Q - R$. Consider a modified formalism that allows the symbol *Z*, distinct from all phonological units and not subsumed under \$, to be referred to in rules. Obtain Y'_i, U'_i, and V'_i from Y_i, U_i, V_i, respectively, by substituting Z^*AZ^* for each phonological unit A. Let $X' = \{X'_1, \ldots, X'_m\}$, where

$$X'_i = Y'_i \rightarrow S_i/U'_i - V'_i \text{ if } Y_i \text{ does not subsume } \emptyset,$$

$$X'_i = \left\{\begin{matrix} \begin{bmatrix} Y'_i \rightarrow S_i/U'_i - V'_i \\ \{Z,\$\}^*\$ \rightarrow \$^*/\{Z,\$\}^* - \{Z,\$\}^* \end{bmatrix} \\ Z \rightarrow S_i/U'_i - V'_i \end{matrix}\right\} \text{ if } Y_i \text{ subsumes } \emptyset.$$

Let X'' be a primitive schema equivalent to X'. The rule $R{:}X''$ has no subrules of the form $\emptyset \rightarrow P/Q - R$ but will simulate N in the following sense. Where N maps a nonempty phonological string $A_1 \ldots A_n$ into $B_1 \ldots B_k$, $R{:}X''$ will map $ZA_1Z \ldots A_nZ$ into $K_1B_1 \ldots K_kB_kK_{k+1}$, where each K_i is a string (possibly empty) of *Z*'s. In the manner described in the preceding paragraph we can construct a left transducer M_1 and a right transducer M_2 such that (M_1, M_2) performs the same mapping as N′. It is a trivial matter to construct a left transducer M_0 that inserts *Z* between each pair of adjacent segments in a nonempty input string and also at the beginning and end of the string. Even more trivial is the construction of a transducer M_3 that deletes all occurrences of *Z*. It is obvious that the 4-tuple (M_0, M_1, M_2, M_3) will provide each nonempty phonological string with the same realization that N does. By virtue of some results of Schützenberger (1961), this 4-tuple is equivalent to a single finite transducer and hence to an ordered pair of machines the first of which is a left transducer and the second a right transducer.*)

In formalizing simultaneous rules we failed to accommodate insertion processes as they are usually formulated. Insertion subrules of right-linear rules are interpreted in the desired way, however. The rule of vowel-doubling proposed for Mohawk by Postal (1969) can, for example, be given as the right-linear rule (45).

(45) $R\colon \emptyset \rightarrow \sigma / \$^*[-\text{syl}] - {}^{ʔ}\begin{bmatrix} +\text{syl} \\ \sigma \end{bmatrix} \$^*$

This will turn *watunisʔas* into *watunisaʔas* by virtue of the application

watunis $_{\text{a}}$ ʔas

This application has the three steps

watunis → watunis/—ʔas
∅ → a/watunis—ʔas
ʔas → ʔas/watunisa—

Left-linear application is the mirror image of right-linear application. Corresponding to the notion of right-overlap is that of left-overlap. If M = P → Q/R — S and N = P′ → Q′ R′ — S′ are two elementary rules we will say that M left overlaps N if and only if

(i) RPS = R′P′S′,
(ii) S is not longer than S′, and
(iii) PS is longer than S′.

Then we formalize left-linear rules as in (46).

(46) (a) A left-linear rule is an expression of the form *L*:X where X is a schema subsuming elementary rules (the subrules of *L*:X) and *L* is a constant denoting the mode of application next to be defined.

(b) Let P and Q be phonological strings and let *L*:X be a left-linear rule. Then *L*:X maps P into Q if and only if there is a sequence $((P_1, Q_1), \ldots, (P_n, Q_n))$ of ordered pairs of strings such that

(i) n is odd;
(ii) $P = P_n \ldots P_1$;

(iii) for each even i, $2 \leq i \leq n-1$, the elementary rule

$$P_i \rightarrow Q_i / P_n \ldots P_{i+1} - Q_{i-1} \ldots Q_1$$

is a subrule of *L*:X;

(iv) for each odd i, $1 \leq i \leq n$, $P_i = Q_i$ and the elementary rule

$$P_i \rightarrow Q_i / P_n \ldots P_{i+1} - Q_{i-1} \ldots Q_1$$

does not left-overlap any subrule of *L*:X;

(v) $Q = Q_n \ldots Q_1$.

The sequence $((P_n, Q_n), \ldots, (P_1, Q_1))$, which we can write as

$$P_n \; \overset{P_{n-1}}{\underset{Q_{n-1}}{}} \; P_{n-2} \ldots \overset{P_2}{\underset{Q_2}{}} \; P_1,$$

will be called an application of *L*:X with input P and output Q. By the ith leftward step of this application we mean the elementary rule

$$P_i \rightarrow Q_i / P_n \ldots P_{i+1} - Q_{i-1} \ldots Q_1$$

The Sanskrit nasal retroflexion can be expressed by a left-linear rule involving the same schema as was used in (42). The left-linear rule gives the same results as the right-linear one, and its application has the same form. For example *uṣnataraanaam* is processed by the application

$$\text{uṣ} \; \overset{\text{n}}{\underset{\text{ṇ}}{}} \; \text{ataraa} \; \overset{\text{n}}{\underset{\text{ṇ}}{}} \; \text{aam}$$

Here, however, the application is regarded as consisting of the following leftward steps:

1. aam → aam/uṣnataraa—
2. n → ṇ/uṣnataraa—aam
3. ataraa → ataraa/uṣn—ṇaam
4. n → ṇ/uṣ—ataraaṇaam
5. uṣ → uṣ/—ṇataraaṇaam

This application bears the same relation to the iterative application (28) as the right-linear application of (42) displayed earlier bears to (27).

By a proof symmetric to that outlined for (43) we can show that for each monogenic left-linear rule N there is a right-transducer M_1 and a left-transducer M_2 such that (M_1, M_2) provides each phonological nonempty phonological string with the same realization that N does.

Let us now review some of the formal results concerning simultaneous and linear rules. We have seen that each monogenic rule that is right-linear, left-linear, or simultaneous is equivalent in the mapping it effects to an ordered pair of machines each of which is right transducer or a left transducer. Hence each such rule is equivalent to some finite transducer in the sense of Schützenberger. By a series of simple constructions, here omitted, we can show converses to these assertions. Specifically, every finite transducer can be simulated by some simultaneous rule.[1] Furthermore, every finite transducer can be simulated by each of at least four ordered pairs of linear rules, representing the four possible combinations of directionality:

1st rule	*2nd rule*
right-linear	right-linear
right-linear	left-linear
left-linear	right-linear
left-linear	left-linear

A further result, due to Schützenberger (1961), is that any n-tuple of finite transducers operating in sequence can be reduced to a single finite transducer. Consequently, if f is a many-to-one mapping of phonological strings into phonological strings, the following assertions are entirely equivalent:

(a) f can be effected by a sequence of finite transducers;
(b) f can be effected by a finite transducer;

[1] It would be absurd, of course, to maintain that the class of phonological rules coincides with the class of rules that can perform finite-state transductions. As the formalism stands it is still possible to formulate many implausible rules, even in a simple way. The extent to which the further necessary restrictions can be stated in general formal terms is not a question we will go into here. It is obvious. however, that substantive contraints, such as those imposed by marking conventions, are indispensable.

(c) f can be effected by a sequence of simultaneous rules;
(d) f can be effected by a simultaneous rule;
(e) f can be effected by a sequence of right-linear rules;
(f) f can be effected by a sequence of left-linear rules.

It makes no difference to mapping capacity, then, which of the three types of rules (simultaneous, right-linear, left-linear) we allow in phonological descriptions. We will therefore have to judge the three types solely on the basis of the formulations they yield. In general, we prefer formulations which reflect naturalness, plausibility, or significant generality with corresponding notational economy. From the point of view of this criterion, it seems clear that we need a formalism that allows both right-linear and left-linear rules, and we shall consider a variety of cases that support this view.

Southern Paiute has a well-known rule which, counting from left to right, stresses the even-numbered nonfinal vowels of a word. If we follow Chomsky and Halle (1968: 244-9) concerning the nature of underlying representations in Southern Paiute, assuming in particular that underlying vowels are all unstressed and are to be equated with Sapir's moras, and if we assume that their rule (44) (which among other things deletes word-final consonants) precedes the alternating stress rule, then we can formulate the latter rule in right-linear fashion as in (47) below. (This formulation is similar to DRULE R7 of Bobrow and Fraser (1968: 769).)

$$(47)\quad R\text{:}\ \begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{stress}\\ (\sigma)\end{bmatrix} /\ \$^{*} \begin{bmatrix}+\text{syl}\\ -\text{stress}\end{bmatrix} [-\text{syl}]^{*} \text{—}\ \$\,\$^{*}$$

To see how this rule works consider the word *natapikkappikai* 'they threw rocks at one another'. The underlying form given here is that presumably imposed by the analysis of Chomsky and Halle. This word is the input of one application of (47), namely:

$$\text{nat}\ \underset{\text{á}}{\text{a}}\ \text{pikk}\ \underset{\text{á}}{\text{a}}\ \text{ppik}\ \underset{\text{á}}{\text{a}}\ \text{i}$$

This application consists of the following rightward steps:

1. nat → nat/—apikkappɨkai
2. a → á/nat—pikkappɨkai
3. pikk → pikk/natá—appɨkai
4. a → á/natápikk—ppɨkai
5. ppɨk → ppɨk/natápikká—ai
6. a → á/natápikkáppɨk—i
7. i → i/natápikkáppɨká—

The even-numbered steps are subrules of (47) and the odd-numbered steps are vacuous and do not right-overlap any subrule of (47). Note that although (47) has the subrules

i → í/natap—kkappɨkai
ɨ → ɨ́/natapikkapp—kai

it will never yield the incorrect stress pattern **natapíkkappɨ́kai* by virtue of any putative application

natap $\genfrac{}{}{0pt}{}{\text{i}}{\text{í}}$ kkapp $\genfrac{}{}{0pt}{}{\text{ɨ}}{\text{ɨ́}}$ kai

The first step here would be

natap → natap/—ikkappɨkai,

which right overlaps the subrule

a → á/nat—pikkappɨkai

The output *natápikkáppɨkái* is subject to further rules which convert it ultimately into *naráẉịkáppɨxáạ*, where . denotes voicelessness. (See Chomsky and Halle, loc. cit. and also Harms 1966b. We assume that the 'spirantization' rule (46) of Chomsky and Halle follows rather than precedes the alternating stress rule; apparently the two rules can occur in either order with no difference in effect.)

Notice that we cannot obtain a simultaneous or left-linear version of the Southern Paiute stress rule merely by substituting *L* or *S* for *R* in (46); if we did we would have a rule that stressed every nonfirst nonlast vowel in a word, deriving forms like **natápíkkáppkíkái.* Apparently the optimum solution in simultaneous or left-linear mode is something like formulation (48).

(48) $L, S\colon \begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{stress}\\ (\sigma)\end{bmatrix} / [-\text{syl}] * [+\text{syl}]$
$([-\text{syl}] * [+\text{syl}]\,[-\text{syl}] * [+\text{syl}]) * [-\text{syl}]* - \$\$*$

This rule states explicitly that a vowel to be stressed must be preceded by an odd-number of vowels within the word, whereas the right linear rule (47) requires merely that the nearest preceding vowel be unstressed. This difference in the way the left-hand context is described is the source of the somewhat greater simplicity of the right-linear formulation.

The case just considered represents a common phenomenon. Eastern Ojibwa, as described by Bloomfield (1956), apparently has an alternating stress rule much like that of Southern Paiute, though with some additional complications. The distinction between underlying long and short vowels, irrelevant in Southern Paiute where phonetic long vowels seem to derive from underlying geminate clusters, is crucial in Eastern Ojibwa. According to Bloomfield (p. 5) the odd-numbered vowels in a sequence containing only short vowels are reduced in "loudness" sometimes to the point of complete disappearance and undergo changes in quality. Other vowels, whether long or short, are not reduced. Furthermore, the last vowel in a word is never reduced. If we interpret reduced loudness as lack of stress and nonreduced loudness as presence of stress, and if we assume underlying vowels to be unstressed, we can formalize Bloomfield's descriptive statement (apart from the quality changes) in terms of the sequence of rules in (49). In (49a) and (49b) any of the three application modes, right-linear, simultaneous, and left-linear, are possible, and we can omit the mode designator.

(49) (a) $\begin{bmatrix}+\text{syl}\\ +\text{tns}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{stress}\\ (\sigma)\end{bmatrix} / \$* - \$*$

(b) $\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{stress}\\ (\sigma)\end{bmatrix} / \$^* \text{—} [-\text{syl}]^*$

(c) R: $\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{stress}\\ (\sigma)\end{bmatrix} / \$^* \begin{bmatrix}+\text{syl}\\ -\text{stress}\end{bmatrix} [-\text{syl}]^* \text{—} \$^*$

(49a) stresses long vowels, (49b) stresses final vowels, and (49c), the only essentially right-linear rule, stresses the even-numbered vowels in a sequence of syllables containing vowels still unstressed (which are identical to nonlast short vowels after application of (49a-b)). We can illustrate the operation of (49) with part of the paradigm of a verb meaning 'to arrive'.

nintakoššin	takoššin	takoššino:k	
		takoššinó:k	48a
nintakoššín	takoššín		48b
nintákoššín	takóššín	takóššinó:k	48c
nentákuššín	tekóššín	tekóššenó:k	reduced-vowel quality changes
'I arrive'	'he arrives'	'they arrive'	

The best we can do in trying to formulate the Eastern Ojibwa alternating stress rule in simultaneous or left-linear mode is apparently (50).

(50) L, S: $\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{stress}\\ (\sigma)\end{bmatrix} / (\$^*[+\text{stress}])^*[-\text{syl}]^*$
$[+\text{syl}, -\text{stress}]([-\text{syl}]^* \begin{bmatrix}+\text{syl}\\ -\text{stress}\end{bmatrix} [-\text{syl}]^*$
$[+\text{syl}, -\text{stress}])^*[-\text{syl}]^* \text{—} \$^*$

(50) says explicitly that a vowel is to be stressed if an odd-number of unstressed vowels intervene between this vowel and the nearest preceding stressed vowel or, if there is no such stressed vowel, the beginning of the word.

Notice that the optimal right-linear versions of the Southern Paiute and Eastern Ojibwa alternating stress rules, (47) and (49c), are almost identical, whereas the optimal simultaneous (and left-linear) versions, (48) and (50), diverge because of the extra

[—stress] specifications necessary in the left-hand context of (50). Thus the right-linear formulations reveal more clearly that the two languages exploit essentially the same alternating stress rule, despite their different treatment of long vowels and final-syllable vowels.

Alternating stress is apparently just one manifestation of a widely attested process which in some sense strengthens or weakens alternate vowels in certain kinds of sequences. We consider briefly three cases in which vowel strength if manifested as length, voicing, and retention, respectively.

In Tübatulabal, according to Swadesh and Voegelin (1939), there is a word-level rule which lengthens the first vowel and each even-numbered vowel in a sequence containing only short vowels.[2] However, a short vowel will not be lengthened if the next following vowel, if any, is long. Some input-output relations defined by this rule are given below. We transcribe the forms according to the principles followed by Swadesh and Voegelin, inferring from one of their suggestions that ∃ (a vowel-shortening variety of glottal stop) is the final segment of the morpheme *la* ∃ 'going'.

dawəginanala∃	adawəginanala∃	(input)
da:wəgi:nana:la∃	a:dawə:gina:nala:∃	(vowel lengthening)
ta:wəgi:nana:la	a:dawə:gina:nala	(later rules)
'he goes along causing him to see'	'he went along causing him to see'	

In Japanese, according to Han (1962: 36-43), alternate vowels are devoiced in a sequence of the form $C_1A_1 \ldots C_nA_nC_{n+1}$ where the C_i are voiceless stops and the A_i are short high unaccented vowels. Whether it is the odd or even vowels that are devoiced is a matter of free variation. Thus we have *pụkupụkuto* and *pukụpukụto* as freely varying pronunciations of *pukupukuto*.

According to Delattre (1951: 348), French has a rule which deletes the even-numbered vowels in a sequence of the form

[2] McCawley (1969) has discussed this rule in great detail and proposed a left-to-right iterative formulation that could be regarded as right-linear.

$C_1A_1 \ldots C_nA_nC_{n+1}$ where the C_i are consonants, A_1 is any vowel, and A_2 through A_n are all schwas. For example, orthographic *elle ne me le redemande pas*, which presumably has the phonological shape *ɛlənəmələrədəmãdəpa* just before the schwa deletion rule is to apply, is pronounced in Delattre's norm as *ɛlnəmlərdəmãdpa*.

The three rules just described bear a striking formal resemblance to left-to-right alternating stress rules and seem, like them, to be most naturally expressed by right-linear formulations.

So far we have considered only rules that affect alternate vowels from left to right. Rules that affect alternate vowels from right to left seem to be somewhat rarer, but they do, in fact, exist, and provide evidence that we must allow left-linear application in phonology. We again consider the Tübatulabal language, which (according to Voegelin 1935: 75-78) has the following block of word stress rules:

(i) the last vowel is stressed,
(ii) each long vowel is stressed, and
(iii) in a sequence containing vowels not stressed by (i) or (ii), the even-numbered vowels, counted from right-to-left, are stressed.

This block of rules is later than the left-to-right alternating length rule that was mentioned above, and indeed follows certain vowel shortening rules which obscure the alternating length pattern. To formalize (i)-(iii) we can look to the very similar Eastern Ojibwa stress rules (49). (i) and (ii) can simply be identified with (49a) and (49b). A formal version of (iii) can be obtained from (49c) by simply replacing *R* with *L* and replacing the contextual portion of (49c) by its mirror image. The stress pattern of Tübatulabal is given, then, by (51).

(51) (a) (like (49a))
(b) (like (49b))
(c) $L\colon \begin{bmatrix} +\text{syl} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{stress} \\ (\sigma) \end{bmatrix} / \$^* \text{—} [-\text{syl}]^* \begin{bmatrix} +\text{syl} \\ -\text{stress} \end{bmatrix} \$^*$

Examples of derivations with (51) are given below.

witaŋhatal	witaŋhatala:batsu	(input)
witaŋhatál	witaŋhatala:batsú	(51a)
	witaŋhatalá:batsú	(51b)
witáŋhatál	wítaŋhátalá:batsú	(51c)

The optimal simultaneous (or right-linear) version of the Tübatulabal alternating stress rule seems to require the much more complex schema (52).

(52) $\begin{bmatrix} +\text{syl} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{stress} \\ (\sigma) \end{bmatrix} / \$^* \text{—} ([-\text{syl}]^* \begin{bmatrix} +\text{syl} \\ -\text{stress} \end{bmatrix} [-\text{syl}]^* \begin{bmatrix} +\text{syl} \\ -\text{stress} \end{bmatrix})^* [-\text{syl}]^* \begin{bmatrix} +\text{syl} \\ -\text{stress} \end{bmatrix} [-\text{syl}]^* [+\text{stress}] \$^*$

Another example of a right to left process affecting alternate vowels is the rule postulated by Havlík (1889) to account for the development of the jers (probably to be interpreted as short high vowels) in Slavic languages. According to this rule, if successive syllables containing jers are counted from the right, then a jer in an even-numbered syllable is retained, though usually changing in quality, while a jer in an odd-numbered syllable is lost. (cf. Shevelov 1965: 452-3). In Russian the retained jers become mid vowels. For example, underlying (or historically earlier) forms *čitici* 'reader nom. sg.' and *čitica* 'reader gen. sg.' yielded *čtec* and *četca*, respectively. Havlík's rule, in its Russian version, can easily be formalized as in (53) if we are permitted left-linear application.

(53) (a) $L\text{:} \begin{bmatrix} +\text{syl} \\ -\text{tns} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} -\text{high} \\ (\sigma) \end{bmatrix} / \$^* \text{—} [-\text{syl}]^* \begin{bmatrix} +\text{syl} \\ +\text{hi} \\ -\text{tns} \end{bmatrix} \$^*$

(b) $\begin{bmatrix} +\text{syl} \\ +\text{hi} \\ -\text{tns} \end{bmatrix} \rightarrow \emptyset / \$^* \text{—} \$^*$

An attempt to formulate (53a) in simultaneous or right-linear fashion will lead to even worse results than in the case of the Tübatulabal alternating stress rule, as can be verified by the diligent reader.

Consider too the rule of Eastern Ojibwa which turns *o* and *i* into *w* and *y*, respectively, before another vowel (Bloomfield 1956: 4-5). As Bloomfield tells us, this rule must be applied from right to left; for example, *eninioak* 'men' becomes *eniniwak*. Simultaneous or left-linear application would sometimes give the wrong result; in particular it would change *eninioak* into **eninywak*. To correct this situation we would have to complicate the rule considerably. The best we would do, it seems, is to say that *o* and *i* become *w* and *y* when occurring before an even number of *o*'s and *i*'s that are followed in turn by a vowel that is preconsonantal, final, or not *o* or *i*.

Certain vowel harmony rules seem also to be best formulated in linear fashion. Consider the rule of Yawelmani Yokuts which rounds a vowel that is preceded by a round vowel of the same height (Kuroda 1967: 13-15, 43-5; Kisseberth 1969). The effects of this rule are exemplified in (54)

(54)	Before harmony	After harmony	Gloss
	hudhin	hudhun	'recognize (aorist)'
	hudal	hudal	'recognize (dubitative)'
	gophin	gophin	'take care of an infant (aorist)'
	gopal	gopol	'take care of an infant (dubitative)'
	muṭmixhin	muṭmuxhun	'swear (comitative aorist)'
	hubuṣxasit	hubuṣxasit	'choose (exclusive passive aorist)'

As can be seen from the next to last example, ronding harmony is propagated as far to the right as possible. We can describe this process quite naturally by means of the right-linear rule (55)

(55) $$R\colon \begin{bmatrix} +\text{syl} \\ \alpha\text{high} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{round} \\ (\sigma) \end{bmatrix} / \$^{*} \begin{bmatrix} +\text{syl} \\ \alpha\text{high} \\ +\text{round} \end{bmatrix} [-\text{syl}]^{*} \text{—} \$^{*}$$

This formulation refers just to the nearest preceding vowel to determine whether rounding of the vowel under consideration should take place, in the spirit of our informal description above. It should be clear that we could not adopt this approach under simultaneous or left-linear application, for each of these types of application considers only the original input form of the left-hand context. Thus we would obtain the incorrect from *muṭmuxhin* from *muṭmixhin*. What we must do when deprived of right-linear application is allow a vowel to be rounded if it is preceded anywhere in the word by a round vowel of the same height, provided that no vowel of a different height intervenes. The necessary schema, given in (56), is much like (55) but requires an additional mention of the feature specification αhigh.

(56) $$\begin{bmatrix}+\text{syl}\\ \alpha\text{high}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{round}\\ (\sigma)\end{bmatrix} / \$^* \begin{bmatrix}+\text{syl}\\ \alpha\text{high}\\ +\text{round}\end{bmatrix} \left[\begin{Bmatrix}-\text{syl}\\ \alpha\text{high}\end{Bmatrix}\right]^* — \$^*$$

From a study of Hetzron's work (1967: 178-9, 193; 1969: 8-9, passim; and personal communication) it appears that Southern Agaw has a right-to-left rule that makes a nonlow vowel high when the next following vowel is *i*. The effect of this rule is illustrated below (tone marks omitted).

gomejanta	gomejanti	moleqeska	moleqesi	(input)
			muliqisi	(raising rule)
(feminine)	(masculine)	(plural)	(singular)	
'one who is in haste'		'monk'		

If my interpretation of this rule is correct, we can express it by means of the left-linear formulation (57).

(57) $$L: \begin{bmatrix}+\text{syl}\\ -\text{low}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{high}\\ (\sigma)\end{bmatrix} / \$^* — [-\text{syl}]^* \begin{bmatrix}+\text{syl}\\ +\text{high}\\ -\text{back}\end{bmatrix} \$^*$$

A simultaneous or right-linear formulation would require us to say that a nonlow vowel is raised when followed anywhere within the word by *i*, provided that no back vowel intervened (underlying front vowels are all nonlow). We would, then need the schema (58), which, though much like the schema in (57), mentions the feature specification — back twice instead of once.

$$(58)\quad \begin{bmatrix} +\text{syl} \\ -\text{low} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{high} \\ (\sigma) \end{bmatrix} / \$^{*} — \left[\left\{\begin{matrix} -\text{syl} \\ -\text{back} \end{matrix}\right\}\right]^{*} \begin{bmatrix} +\text{syl} \\ +\text{high} \\ -\text{back} \end{bmatrix}$$

Consonants as well as vowels may be subjected to processes of a linear character. We mention first a rather simple and commonplace example. In Russian and certain other languages an obstruent cluster becomes voiced or voiceless throughout, according as the final member of the cluster is voiced or voiceless. Thus Russian *ksgibu* 'toward the bend' is actually pronounced *gzgibu*. The left-linear formulation of the rule, given in (59) seems superior to the optimal simultaneous or right-linear formulations (60).

$$(59)\quad L\text{:}\ \begin{bmatrix} -\text{son} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} \alpha\text{voice} \\ (\sigma) \end{bmatrix} / \$^{*} — \begin{bmatrix} -\text{son} \\ \alpha\text{voice} \end{bmatrix} \$^{*}$$

$$(60)\quad S,R\text{:}\ \begin{bmatrix} -\text{son} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} \alpha\text{voice} \\ (\sigma) \end{bmatrix} / \$^{*} — [-\text{son}]^{*} \begin{bmatrix} -\text{son} \\ \alpha\text{voice} \end{bmatrix} \left\{\begin{matrix} [+\text{son}]\$^{*} \\ \emptyset \end{matrix}\right\}$$

The schema in (60) must explicitly identify the final member of the obstruent cluster as the agent determining the voicing of the obstruent under consideration, whereas the schema in (59) needs mention only the immediately following obstruent.

Another example of a linear process affecting consonants is to be found in Tshiluba.[3] Consider the verbal suffixes *-il-* (benefactive) and *-ile* (simple past). We illustrate their use first with the root *-kwat-* 'take'.

kukwata	'to take'
ukwačile	'he took'
kukwačila	'to take (ben.)'
ukwačid^{y}ile	'he took (ben.)'

[3] Our statements about Tshiluba are based largely on work with a native informant, M. Pierre Mulumba. This work was carried out during the academic year 1969-70 at the University of California, Santa Barbara, under the auspices of the Linguistics Program. For further information about Tshiluba see Burssens (1946) and Coupez (1954).

The changes *t* to *č* and *l* to *d*^{y} before *i* are quite regular. Consider now the parallel paradigmatic forms of the verb -d^{y}im- 'cultivate'.

kud^{y}ima	'to cultivate'
ud^{y}imine	'he cultivated'
kud^{y}imina	'to cultivate (ben.)'
ud^{y}imin^{y}ine	'he cultivated (ben.)'

The rule is that *l* becomes *n* when the nearest proceding consonant is nasal, with *n* later becoming palatalized before *i*. The form *ud^{y}imin^{y}ine*, derived from *ud^{y}imilile*, indicates that the rule proceeds from left to right. Since *n* and *l* are the only coronal sonorants in Tshiluba we can give the rule in a right-linear formulation as follows:

$$(61) \qquad R: \begin{bmatrix} +\text{son} \\ +\text{cor} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{nas} \\ (\sigma) \end{bmatrix} / \$^*[+\text{nas}]\,[+\text{syl}] - \$^*$$

In a simultaneous or left-linear formulation we would have to write (62).

$$(62) \qquad S, L: \begin{bmatrix} +\text{son} \\ +\text{cor} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{nas} \\ (\sigma) \end{bmatrix} / \$^*[+\text{nas}](\,[+\text{syl}] \begin{bmatrix} +\text{son} \\ +\text{cor} \end{bmatrix}\,)^* - \$^*$$

The complication arises from the fact that we must refer to the left context in its original input form (e.g. ud^{y}imili— rather than ud^{y}imini—). Thus we must state explicitly that *l*'s can intervene between the *l* being considered and the preceding *n* that would trigger the nasalization. In right-linear application the intervening *l*'s will have already become *n*'s themselves, so to speak, so that the environment needn't be complicated to account for them.

The evidence considered so far favors a formalism that allows both right-linear and left-linear rules. With these types of rules available we might naturally inquire whether we need simultaneous ones as well. Apparently the answer is no, for it is difficult to find a genuine case where simultaneous mode of application yields a better formulation than either of the linear modes. We can, however, get an idea of what such a case might be like by considering the effects of the second singular prefix in the Terena language (Bendor-Samuel 1960 and Langendoen 1968: 109-10). According to Langendoen's interpretation of Bendor-Samuel's work, this prefix has the underlying phonological shape *y*. When attached to a stem beginning with a vowel, no further change takes place. On the other hand, if the *y* is prefixed to a form beginning with a consonant, then the first non-*i* vowel undergoes a change, as follows: *u* and *e* become *i*, and *a* and *o* become *e*. Subsequently the *y* is deleted. Thus we have such forms as the following:

nokone	'he needs'	nekone (<ynokone)	'you need'
kurikena	'his peanut'	kirikena (<ykurikena)	'your peanut'
piho	'he went'	pihe (<ypiho)	'you went'

What we have just described is the main clause of the *y*-prefix rule. There is in addition a special clause that raises an *e* to *i* in a *y*-prefixed word if all the other vowels in the word are also *e*'s. The special clause can be thought of a preceding the main clause. An example is *yxerere* 'your side', which becomes *yxiriri* by the special clause (ultimately the *y* is deleted). On the other hand, a word of the form *yBeCeDi*, where B, C, and D are consonants, would supposedly be untouched by the special clause because not all its vowels are *e*'s. The word would then pass on to the main clause, becoming *yBiCeDi*. Similarly, *yBiCeDe* would be unaffected by the special clause and would consequently become *yBiCiDe* by the main clause. It seems that only a simultaneous formulation of the special clause would be natural, since it is apparently the original input form of both the left and right contexts that determines whether an *e* will be raised. Linear application, which requires us to refer to the output form of either the left or the right context, would force us to express the special clause in a more roundabout way. Perhaps the best we could do is introduce the following two rules, each of which could be left-linear or right-linear.

(i) Change *e* to E in a *y*-prefixed word when every vowel to the left or right is *e* or E;
(ii) Change E to *i*.

We assume here that E is some vowel that does not otherwise occur in Terena.

The special clause of the *y*-prefix rule of Terena is not in fact a clear case of an essentially simultaneous rule. There is a subtle but crucial difference between Langendoen's description of the rule and Bendor-Samuel's 1960 statement. What Bendor-Samuel actually says is that if the first one or more vowels of a word are all *e*'s, then these *e*'s are all raised to *i*. This implies, apparently, that we need only refer to the left context of an *e* to determine whether the *e* should be raised to *i*. Thus *yBeCedi* would indeed be affected, becoming *yBiCidi*. It is difficult to determine from Bendor-Samuel's data whether this is true; if it is, a left-linear formulation will be completely adequate.

Although the Terena example fails to provide us with a convincing case of simultaneity, it does present us with another type of rule, not hitherto discussed, which is interesting to take note of. Consider (63), which in its environment part is a near optimal formulation of the main clause of the Terena *y*-prefix rule.

$$(63)\quad \left\{\begin{matrix} \begin{bmatrix} +\text{syl} \\ +\text{back} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} -\text{low} \\ -\text{back} \\ (\sigma) \end{bmatrix} \\ \begin{bmatrix} +\text{syl} \\ -\text{back} \\ \sigma \end{bmatrix} - \begin{bmatrix} +\text{high} \\ (\sigma) \end{bmatrix} \end{matrix}\right\} \Big/ \begin{bmatrix} -\text{cons} \\ -\text{syl} \\ -\text{back} \end{bmatrix} [-\text{syl}] \left\{\begin{matrix} [-\text{syl}] \\ \begin{bmatrix} -\text{back} \\ +\text{high} \end{bmatrix} \end{matrix}\right\}^{*} — \$^{*}$$

Right-linear application cannot be associated with this schema because it would change *ykurikena* to **kirikine* instead of the correct *kirikena*. However, both left-linear and simultaneous application give the correct results. The reverse situation is exemplified by Dahl's law, a rule occurring in a number of Bantu languages of East Africa (Bennett 1967). The Southern Kikuyu version of this rule causes a *k* to become *γ* when the next following consonant in the word is an underlying voiceless stop. Any number or vowels may intervene between the *k* and the voiceless stop. As so stated, the rule can apply either simultaneously or right-linearly, producing, for example, *neγaγaakeroma* from *nekakaakeroma* 'he will bite him'. (I am indebted to Leonard Talmy for this example.) A left-linear application would yield incorrectly the form **nekaγaakeroma*, and consequently some other (and more complicated) means of describing the change would be necessary.

To summarize, we have found a fair number of rules that favor a right-linear formulation and a fair number that favor a left-linear formulation. In addition we have given an example of a rule that can be regarded equally well as left-linear or simultaneous, though not plausibly as right-linear, and another example that can be regarded as right-linear or simultaneous though not plausibly as left-linear. Then, of course, there are many rules (perhaps the majority) which are simply indifferent as to mode of application, working equally well in right-linear, simultaneous, and left-linear mode. The Sanskrit nasal retroflexion rule is of this kind. However, we have failed to find a convincing example of an exclusively simultaneous rule. These results suggest that we allow only right-linear and left-linear rule application in phonology.

6

ALTERNATIVES TO LINEAR RULES

A right linear rule moves inexorably rightward through an input string. Having changed a segment, a right linear rule moves on, never changing that segment again nor changing anything to the left of that segment. Although this is one plausible way of formalizing left-to-right processing, it is not the only way, and it might be wrong. We might have been better advised to define a restricted iterative rule that changes only the leftmost possible segment at each step of application. Such a rule we might call left-iterative, representing it in the form *LI*:X, where X is a schema subsuming elementary rules. To achieve the effect of leftmost application formally, we might say that $P \rightarrow Q/R - S$ is a subrule of *LI*:X if and only if X subsumes $P \rightarrow Q/R - S$ but subsumes no elementary rule $P' \rightarrow Q'/R' - S'$ where $R'P'S' = RPS$ and R' is shorter than R. We might further require that $P \neq Q$ in order to exclude vacuous subrules. We would then say that *LI*:X maps U into V if and only if there is a sequence $U_1, \ldots, U_n$ of phonological strings such that

(i) $U = U_1$,
(ii) for each i, $1 \leq i \leq n - 1$, some subrule of *LI*:X has U_i as input and U_{i+1} as output,
(iii) U_n is not the input to any subrule of *LI*:X, and
(iv) $U_n = V$.

The Southern Paiute stress rule, for example, could just as well have been regarded as left-iterative, having the form of (64).

(64) *LI*: $\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}+\text{stress}\\ (\sigma)\end{bmatrix} / \$^{*} \begin{bmatrix}+\text{syl}\\ -\text{stress}\end{bmatrix} [-\text{syl}]^{*} - \$ \$^{*}$

This rule will give the correct stress pattern to *natapikkappɨkai* by virtue of the following application:

natapikkappɨkai	
natápikkappɨkai	(by subrule a → á/nat—pikkappɨkai)
natápikkáppɨkai	(by subrule a → á/natápikk—ppɨkai)
natápikkáppɨkái	(by subrule a → á/natáppikkáppɨk—i)

Notice that the elementary rule

i → í/natap—kkappɨkai,

which would put a stress in the wrong place, is not a subrule of (62) even though it is subsumed under the schema in (62). The reason becomes apparent when we set

R = natap,	R′ = nat
P = i,	P′ = a,
S = kkappɨkai,	S′ = pikkappɨkai
Q = í,	Q′ = á.

Then the elementary rule under consideration will be P → Q/R — S. However, the schema in (62) also subsumes P′ → Q′/R′ — S′, where RPS = R′P′S′ and R′ is shorter than R; this situation prevents P → Q/R — S from being a subrule of (62) according to the definitions in the preceding paragraph. Note also that á → á/nat — pikkáppɨkái cannot be a subrule of (62) because it is vacuous; consequently the application displayed above is complete and does not continue indefinitely.

In parallel fashion we could define a notion of right-iteration, to take the place, perhaps, of left-linear application.

Although the linear rules discussed in the preceding chapter would work as well in left- or right-iterative fashion, we have some slight reason to prefer the linear formalization. First of all, left or right iteration allows for some peculiar effects going beyond the bounds of finite-state processing, effects which we will probably want to exclude. For example, rule (65) will convert each string of the form $PCAA_1...A_nB_1...B_mB$, where C, B_1, ..., B_m. B are consonants and A, A_1, ..., A_n are vowels, into $PCAA_1...A_{n-m}B$ or $PCAB_{n+1}...B_mB$ according as $n \geqq m$ or $m > n$.

(65) *LI*: $[+\text{syl}]\,[-\text{syl}] \rightarrow \emptyset / \$^* [+\text{syl}] - [-\text{syl}]\ ^*\ [-\text{syl}]$

The right and left iterative modes of application would have the further disadvantage of forcing a third mode of application on us. To see this, consider first any rule which switches the value of some feature, such as the first part of the English vowel shift as described by Chomsky and Halle. This rule can be given as

(66) $$\begin{bmatrix} +\text{syl} \\ +\text{tns} \\ +\text{stress} \\ -\text{low} \\ \alpha\text{high} \end{bmatrix} \rightarrow \begin{bmatrix} -\alpha\text{high} \\ (\sigma) \end{bmatrix} / \ \$^* - \$^*$$

The absence of mode designator is intended to mean that the rule can be applied in either right-linear or left-linear fashion (in fact, it can also be applied simultaneously). Either way the rule will convert *kū́w* 'cow' into *kū́w* (ultimately *káw*), by virtue of an application having the form k $\overset{\acute{\bar{u}}}{\underset{\acute{\bar{o}}}{}}$ w. Suppose, however, that (66) were left-iterative or right-iterative. Then it would keep on applying to *kū́w* forever, changing it first to *kṓw*, then back to *kū́w*, then to *kṓw* again, and so on. To make (66) work properly we would have to introduce some third type of application, the most natural choice being simultaneous.

Again, consider the rule of Chipewyan which devoices a continuant consonant that immediately follows a voiceless continuant consonant (Li 1946: 400). The rule causes *tɛszáih* 'I split' to become *tɛssáih* (ultimately *tɛsáih*). It will also cause *náslzɛ́* 'I am hunting' to become *násɬzɛ́* (ultimately *nászɛ́*). Notice that *náslzɛ́* will not become **násɬsɛ́*; it is the immediately preceding sound in the original input that determines whether devoicing of a continuant consonant will take place. The appropriate effect can be achieved by associating simultaneous or left-linear application, though not right-linear application, with the following schema:

$$\begin{bmatrix} -\text{syl} \\ +\text{cont} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} -\text{voice} \\ (\sigma) \end{bmatrix} / \$^* \begin{bmatrix} -\text{voice} \\ +\text{cont} \end{bmatrix} - \$^*$$

On the other hand, both left-iterative and right-iterative application of this schema would turn *náslzé* incorrectly into **násɬsé*. Again, some third mode of application would have to be introduced to account for this case correctly.

Some of the defects of left and right iteration could be cured as follows. We might say that a subrule $P \rightarrow R/Q - S$ changes RPS into $R < Q > S$, where P contains no angle brackets. After application of a rule is completed, angle brackets are erased. The effect, roughly, is to exclude a substring from further change if it has already been changed. By this of mechanism we could exclude at least some of the nonfinite-state effects referred to above (in particular, rule (65) would now turn $PCAA_1 \ldots A_nB_1 \ldots B_mB$ into $PCAA_1 \ldots A_{n-1}B_2 \ldots B_mB$), and we could make feature-switching rules function properly. However, the Chipewyan rule would still work improperly under both left and right iteration. To correct this situation we could attempt some further refinements of left and right iteration, but it seems pointless to do so when linear application accounts correctly for all the cases discussed.

Anderson (1968) has proposed a convention of left-to-right application that is rather different from either right-linear application or left-iteration. According to Anderson's proposal we must think not of left-to-right application of a single rule but rather of a sequence of rules. When application is to begin a position marker $^{\text{o}}$ is placed immediately to the left of the first vowel in the input form. The rules, each of which contains one mention of the position marker in its environment, are then applied in sequence in the usual fashion. After the sequence is applied, the symbol $^{\text{c}}$ is moved rightward until it is at the immediate left of the second vowel in the string, and the rule sequence is applied again. This procedure is repeated as long as possible, and is therefore executed as many times as there are vowels in the input string. If we identify vowels with syllables, or somehow restrict the placing of the position marker so that it appears only to the immediate left of syllable peaks, we can refer to this mode of application as the left-to-right syllabic cyclic.

Let us write a left-to-right syllabic cycle in the form *LRSy*: $(X_1, \ldots, X_n)$ where each X_i is a schema subsuming elementary rules. Application of the cycle might then be formalized as follows. Let P and Q be phonological strings. Then *LRSy*: $(X_1, \ldots, X_n)$ maps P into Q if and only if there is a sequence

$$\mathscr{S} = ((R_{1,1}, \ldots, R_{1,n+1}), \ldots, (R_{m,1}, \ldots, R_{m,n+1}))$$

of (n+1)-tuples of strings such that

(i) P is mapped into $R_{1,1}$ by the rule
S: $[+\text{syl}, \sigma] \rightarrow {}^{\circ}\sigma/[-\text{syl}]^{*} - \$^{*}$

(ii) For each i, $1 \leqq i \leqq m$, and each j, $1 \leqq j \leqq n$, the rule
S: X_j maps $R_{i,j}$ into $R_{i,j+1}$;

(iii) For each i, $1 \leqq i \leqq m-1$, the string $R_{i,n+1}$ is mapped into $R_{i+1,1}$ by the rule
$$S: \begin{Bmatrix} {}^{\circ} \rightarrow \varnothing/\$^{*} - \$^{*}, \\ [+\text{syl}, \sigma] \rightarrow {}^{\circ}\sigma/\$^{*}\ {}^{\circ}[+\text{syl}]\ [-\text{syl}]^{*} - \$^{*} \end{Bmatrix}$$

(iv) $R_{m,n+1}$ is mapped into Q by the rule S: ${}^{\circ} \rightarrow \varnothing/\$^{*} - \$^{*}$.

The sequence $\mathscr{S}$ is said to be an application of *LRSy*: $(X_1, \ldots, X_n)$ with input P, and the (n+1)-tuple $(R_{i,1}, \ldots, R_{i,n+1})$ is referred to as the ith round or cycle of the application.

A very simple example of a left-to-right syllabic cycle is (67).

(67) $$LRSy: \left(\begin{bmatrix} +\text{syl} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{stress} \\ (\sigma) \end{bmatrix} / \$^{*} \begin{bmatrix} +\text{syl} \\ -\text{stress} \end{bmatrix} [-\text{syl}]^{*} - \$\$^{*} \right)$$

This cycle contains just one rule, whose schema is identical (except for the symbol °) with that used in (47). In fact, (67) produces the same results as (47) and is yet another way of describing the Southern Paiute stress pattern. (67a) below is an application of

(67) to the word *tuk*^{w}*apaiyu* 'during the night'. Since (67) contains only one rule, each round of (67a) is a 2-tuple of strings.

(67a)	Round 1:	(t^{o}uk^{w}apaiyu,	t^{o}uk^{w}apaiyu)
	Round 2:	(tuk^{wo}apaiyu,	tuk^{wo}ápaiyu)
	Round 3:	(tuk^{w}áp^{o}aiyu,	tuk^{w}áp^{o}aiyu)
	Round 4:	(tuk^{w}ápa^{o}iyu,	tuk^{w}ápa^{o}íyu)
	Round 5:	(tuk^{w}ápaíy^{o}u,	tuk^{w}ápaíy^{o}u)

In this application, as in all applications of (67), the odd-numbered rounds are vacuous. Usually we will display applications of left-to-right cycles in a completely vertical manner, omitting vacuous steps. Thus (67a) would appear as follows:

t^{o}uk^{w}apaiyu
tuk^{wo}apaiyu
tuk^{wo}ápaiyu
tuk^{w}áp^{o}aiyu
tuk^{w}ápa^{o}iyu
tuk^{w}ápa^{o}íyu
tuk^{w}ápaíy^{o}u

Anderson does not simply propose that left-to-right syllable cycles be made available along with other more orthodox types of application. He claims at one point that phonologies of natural languages consist entirely of simultaneous rules and left-to-right syllabic cycles. To assess the strength and validity of this claim, let us first consider left-to-right syllabic cycles from the point of view of mapping power. If we assume that a LRSy cycle can contain any rule schema that subsumes elementary rules of the form $P \rightarrow Q/R^{o}S - T$ or $P \rightarrow Q/R - S^{o}T$ and that each such schema is to be applied in simultaneous mode, then we have a rule formalism that can simulate any simultaneous rule and hence any sequence of simultaneous rules. For suppose we have a single simultaneous rule N. We can assume that N has the form S: $\{A_1 \rightarrow Q_1/X_1 - Y_1, \ldots, A_n \rightarrow Q_n/X_n - Y_n\}$ where the A_i are phonological units, the Q_i are phonological strings, and the X_i and Y_i are primitive schemata subsuming phonological strings.

For each i, let X'_i and Y'_i be obtained from X_i and Y_i, respectively, by replacing each phonological unit B with the expression $(^\circ)\ {}^*B(^\circ)^*$. Let Z be the schema

$$\left\{\begin{array}{l} [-\text{syl}]^* \rightarrow \$^*/\$^{*\circ}[+\text{syl}]\,[-\text{syl}]^*\text{—} \\ \$^* \rightarrow \$^*/\$^*\left\{\begin{array}{l} {}^\circ\text{—} \\ \text{—}\,\$^{*\circ}[+\text{syl}] \end{array}\right\} \end{array}\right\}[-\text{syl}]^*$$

Then the left-to-right syllabic cycle

$$LRSy\text{: } ([\{A_1 \rightarrow Q_1/X'_1\text{—}Y'_1,\ \ldots,\ A_n \rightarrow Q_n/X'_n\text{—}Y'_n\},Z])$$

which contains just one rule, albeit a complex one, will accomplish the same task as the rule N. The bracketed expression can, of course, be eliminated in favor of a primitive schema. The idea behind this construction is that the left-to-right syllabic cycle should simply idle until it reaches the last round. On the last round, the lone rule in the cycle is executed and application terminates.

With left-to-right syllabic cycles we can also carry out certain nonfinite-state processes. Consider cycle (68) which contains just one rule.

(68) $LRSy\text{: } ([-\text{syl}] \rightarrow \varnothing/[-\text{syl}][-\text{syl}]^*\text{—}[+\text{syl}]^{*\circ}[+\text{syl}]\,\$^*)$

This cycle turns a string of the form $CC_1 \ldots C_nA_1 \ldots A_mP$, $(n \geqq 0, m \geqq 1)$, where $C, C_1, \ldots, C_n$ are consonants, $A_1, \ldots, A_m$ are vowels, and P is empty or begins with a consonant, into $CA_1 \ldots A_mP$ or $CC_1 \ldots C_{n-m}A_1 \ldots A_mP$ according as $m \geqq n$ or $m < n$. We illustrate with applications of (68) to strings cccccaaac and ccaaac.

ccccc$^\circ$aaac	cc$^\circ$aaac
cccc$^\circ$aaac	c$^\circ$aaac
cccca$^\circ$aac	ca$^\circ$aac
ccca$^\circ$aac	
cccaa$^\circ$ac	caa$^\circ$ac
ccaa$^\circ$ac	

The mapping power of left-to-right syllabic cycles might be reduced if a different method of applying rule schemata were stipulated. Certain strong constraints on the form of rule schemata might also have this effect, and it is possible that we have done an injustice to Anderson's theory by not imposing such constraints. However, Anderson is silent on the matter.

Although left-to-right syllabic cycles probably have sufficient mapping capacity for phonological purposes, perhaps even too much mapping capacity, they are clearly incapable of providing linguistically satisfactory formulations in many cases. One of the difficulties arises from the fact that Anderson excludes right-to-left cycles because of his mistaken belief that right-to-left processes do not occur in the phonologies of natural languages. Thus, although the right-linear rules discussed in the preceding chapter can be routinely reformulated as left-to-right syllabic cycles (we need only place a position marker to the left of each environment dash), the left-linear rules would have to be formulated with essentially the same complex and unsatisfactory schemata that are needed in the simultaneous formulations.

Suppose we extend Anderson's formalism so as to allow right-to-left syllabic cycles as well as left-to-right ones. We would still be unable to express the right-to-left nature of the Russian rule that determines the voicing of obstruent clusters according to the final member of the cluster (rule (59) in the preceding chapter). The problem here is that the right-to-left processing takes place within each cluster, proceeding segment by segment rather than syllable by syllable. We can, however, amend Anderson's theory again so that the cycles are segmental rather than syllabic. Thus the position marker will now be placed at the beginning (or end) of the input string when application is about to begin and will be moved one segment (or, more generally, one phonological unit) to the right (or left) at the end of each round. We then will have the following two types of cycles:

LR	(left-to-right segmental)
RL	(right-to-left segmental)

All the left-to-right syllabic cycles discussed above, including the nonfinite-state cycle (68), will work properly as left-to-right segmental. The same is true of the cycles formulated by Anderson, which will be discussed below.

Anderson makes an even stronger claim than the one that we have been discussing. He believes that every phonology consists of exactly one left-to-right cycle. It should be clear by now that this claim is untenable. Note that we cannot even say that each particular language is characterized by a single direction, having either all left-to-right cycles or all right-to-left cycles. In the preceding chapter we mentioned at least two languages, Tübatulabal and Eastern Ojibwa, that had both left-to-right and right-to-left rules. Tübatulabal as a left-to-right rule of alternating vowel length and a right-to-left stress rule, while Eastern Ojibwa has a right-to-left rule making glides out prevocalic short nonlow vowels and a left-to-right stress rule.

Although most of Anderson's claims appear to be wrong, it still might be true that left-to-right (right-to-left) processing is more correctly formalized as right-to-left (left-to-right) segmental cyclic rather than left (right) linear. Let us review some Finnish evidence, presented by Anderson, which seems superficially to support this view.

Finnish[1] has a consonant gradation rule that weakens a single or geminate stop that follows a sonorant and begins a closed syllable. The standard results of weakening are indicated below.

p becomes v,
t becomes d,
k is deleted,
a geminate stop becomes a single stop.

Thus we have forms such as the following:

[1] Among the many works about Finnish in a well-known language are Fromm and Sadeniemi (1956), Harms (1964), McCawley (1963, 1966), Lehtinen (1967). Our assertions about the language are based mainly on work with Miss Sointu A. Takala, who served as informant in a field methods course offered in the Linguistics Department of the University of California, Berkeley, during the academic year 1964-65.

matto 'rug (nom.)'
maton 'rug (gen.)'
mato 'worm (nom.)'
madon 'worm (gen.)'

The allative plural of *tyttö* 'girl' shows that a vowel $+i$ cluster must be regarded as tautosyllabic for the purposes of consonant gradation:

tyttöille (before gradation)
tytöille (after gradation).

There is another rule which deletes any *t* or *d* that follows a non-initial [−syl] [+syl] sequence and precedes another vowel. Some effects of this rule are seen in the partitive suffix *ta*:

maa	'land (nom.)'
maata	'land (part.)'
talo	'house (nom.)'
taloa	(<talota) 'house (part.)'

Suppose now we attempt to order dental-stop deletion and consonant gradation in the conventional manner. On the one hand we might decide that dental-stop deletion precedes consonant gradation, in view of the fact that this ordering yields the correct derivation for the genitive singular of *ammatti* 'occupation':

ammatin
ammattin (dental-stop deletion)
ammatin (consonant gradation)

The opposite ordering would incorrectly give **ammain*, as follows:

ammattin
ammatin (consonant gradation)
*ammain (dental-stop deletion)

On the other hand, consider the so-called collective genitive plural of *harakka* 'magpie'. If consonant gradation precedes dental-stop deletion, then the correct derivation of the form is obtained:

harakkatin
harakkadin (consonant gradation)
harakkain (dental-stop deletion)

The other ordering, which seems to be necessary for the genitive singular of *ammatti*, yields the wrong result for the collective genitive plural of *harakka*:

harakkatin
harakkain (dental-stop deletion)
*harakain (consonant gradation)
*harakoin (other rules).

Anderson chooses to resolve this paradox by means of a left-to-right syllabic cycle. The cycle he proposes can just as well be regarded as segmental and given roughly in the form of (69). (69a) presupposes a rule that stresses the first and only the first vowel of a word. Just how the portion of (69b) to the left of the slash is to be adequately expressed is not clear; this matter will be touched on later.

(69) *LR*:

(a) Dental-stop deletion.

$$\begin{bmatrix}-\text{cont}\\+\text{cor}\end{bmatrix} \rightarrow \varnothing / \$^{*}\,[-\text{syl}]\begin{bmatrix}+\text{syl}\\-\text{stress}\end{bmatrix} \text{—}^{o}[+\text{syl}]\$^{*}$$

(b) Consonant gradation.

$$\left\{\begin{matrix}\text{tt}\rightarrow\text{t}\\ \text{t}\rightarrow\text{d}\\ \cdot\\ \cdot\\ \cdot\end{matrix}\right\} / \$^{*}[+\text{son}]\text{—}^{o}[+\text{syl}]\begin{Bmatrix}\text{i}\\ \varnothing\end{Bmatrix}[-\text{syl}]\begin{Bmatrix}[-\text{syl}]\$^{*}\\ \varnothing\end{Bmatrix}$$

The correct form of the words considered above will now result, as can be seen from the following partial derivations:

(i) ammatt^{o}in
ammat^{o}in (69b)
ammati^{o}n

(ii) harakk^{o}atin
harakka^{o}tin
harakkat^{o}in
harakka^{o}in (69a)
harakkai^{o}n

Though Anderson's artifice is clever, the need for it is highly questionable. Notice, first of all, that conventional rule ordering can be imposed if we split the gradation rule into at least two stages. The first stage would be much like the gradation rule in (69), except that it would merely weaken one half of a geminate stop rather than eliminating that half altogether. In other words, the gradation rule proper would change *tt* into *t̆t*, say, where *t̆* is some weakened variety of *t*, and a later rule following dental-stop deletion, would reduce *t̆t* to *t*. We would then have derivations such as the following:

ammattin	harakkatin	
ammat̆tin	harakkadin	(consonant gradation)
ammat̆tin	harakkain	(dental-stop deletion)
ammatin	harakkain	(weak geminate reduction)

Despite Anderson's assertion to the contrary, there is good independent evidence that geminates pass through an intermediate stage when they are subject to gradation. To see that this is so we will have to consider a number of rules which appear at first glance to be irrelevant to the present problem.

First, although the usual results of weakening *p*, *t*, *k* in the gradation environment are *v*, *d*, and ∅, respectively, special circumstances will call forth a different output. For example, if the sonorant preceding the stop is a homorganic nonsyllabic, the stop assimilates completely to the sonorant; for example, underlying *lukenton* (genitive singular of *lukento* (> *luento*) 'lecture') becomes *luennon*. Another fact, of particular interest here, is that *k* becomes *j* in the gradation environment when the preceding sonorant is *r*, *l*, or *h* and the following vowel is *e*. Thus form the verb stem *särke-* 'break' we have first person singular present *särjen* (cf.

third person plural present *särkevät*). It should be borne in mind that it is only the vowel *e* which triggers the change *k* to *j*; even the closely related vowel *i* will not do it. Thus when the first singular suffix *n* is appended to the verb stem *pyrki-* 'strive for' the phonetic form *pyrin* results according to the general rule, not *pyrjin*.

Consider next the rule that deletes a short unrounded vowel when a suffixal *i* follows. The effects of this rule can be illustrated by the verb stems *pyrki-*, cited above, *hake-* 'look for', and *otta-* 'take'. If the third person suffix *vat* alone is added to these stems, the resulting forms are simply *pyrkivät* 'they strive for' (with vowel harmony in the suffix), and *hakevat* 'they look for', *ottavat* 'they take'. If the past tense suffix *i* is inserted too, giving the underlying strings *pyrkiivät*, *hakeivat*, and *ottaivat*, the phonetic results will be *pyrkivät*, *hakivat*, and *ottivat*. If we add the past and first singular suffixes to the stems in question, obtaining the underlying forms *pyrkiin*, and *hakiin*, and *ottiin*, both gradation and unround-vowel-deletion will apply, yielding phonetic *pyrin*, *hain*, and *otin*.

A verb stem ending in *rke*, *lke*, or *hke*, will of course also be subject to unround-vowel-deletion. The crucial question is what happens when both unround-vowel-deletion and gradation apply. Applied in the order just stated, the *k* would be deleted; thus the first singular past of *särke-* would be obtained by the following derivation:

särkein	
särkin	(unround-vowel-deletion)
särin	(gradation)

The output here, however, is incorrect; the form actually used is *särjin*. Consequently, consonant gradation must precede unround-vowel-deletion, and we must have the derivation

särkein	
särjein	(gradation)
särjin	(unround-vowel-deletion)

Another rule we must consider is the one that changes noninitial *t* to *s* before *i*. This "dental-stop sibilation" rule, together with a rule that raises final *e* to *i*, accounts for nominative singular *käsi* 'hand' in face of the essive singular *kätenä*, genitive *käden* (with gradation), and so on. It also accounts for the past tense forms of the many verb stems that end in *t*. An example is *halut-* 'want' with third singular imperative *halutkoon*, infinitive *haluta* (from *haluttah* by consonant gradation and final *h* deletion), first singular past *halusin* (< *halutin* by sibilation). The last form indicates that dental-stop deletion must follow sibilation; otherwise we would obtain **haluin*. The stem *kiipet-* 'climb' behaves in a similar manner; we have *kiivetköön* (< *kippetkoon*), *kiivetä* (< *kiipettah*), *kiipesin* (< *kiipetin*).

Consider now a verb stem like *tunte-*, 'feel' infinitive *tuntea* from underlying *tuntetah*. In the present we have first singular *tunnen*, from *tunten* by consonant gradation, and third plural *tuntevat*. In the past we have first person singular *tunsin* and third plural *tunsivat*, from underlying *tuntein* and *tunteivat*. Clearly, unround-vowel-deletion must precede sibilation of *t* if these forms are to be derived correctly.

The data we have considered so far imposes the following order on the rules under discussion:

consonant gradation
unround-vowel-deletion
sibilation of dental stops
dental-stop deletion

If we accept this ordering, which seems to be unavoidable even in a left-to-right cycle, we must perforce accept an intermediate stage in the gradation of geminate stops. For if the gradation rule immediately reduced *tt* to *t* then the rules given above would turn *ammattin* into **ammasin* rather than the correct form *ammatin*. Gradation must, then, weaken *tt* to some intermediate form which cannot be reduced to *t* until after both dental-stop sibilation and dental-stop deletion have applied. The precise nature of this intermediate stage is not altogether clear, to be sure; its historical analogue is denoted *t̆t* in the handbooks (cf. Fromm and Sadeniemi 1956: 35-6) in accordance with a hypothesis that the first part of the cluster was weakened (laxed?). Nevertheless, the basic point remains. If geminate stops must pass through an intermediate stage anyway when subject to gradation, a major motivation for the left-to-right cycle vanishes.

Note that geminate stops are not the only clusters that must pass through an intermediate stage under the rule ordering that seems to be necessary. Consider again the first singular present and past of *tunte-*. We saw above that these forms had the underlying representations *tunten* and *tuntein*. The phonetic realization of *tunten* is *tunnen*, which arises straightforwardly by gradation. The phonetic realization of *tuntein*, however, is *tunsin*, which must have undergone dental-stop sibilation. But if gradation changes *nt* immediately to *nn*, our rules give the following incorrect derivation:

tuntein	
tunnein	(gradation)
*tunnin	(unround-vowel deletion)
	(no further rules applicable)

Apparently, then, we must assume that a single stop in gradation environment is first weakened by some simple operation such as laxing or voicing, regardless of what the preceding sonorant is. This weakened stop is still subject to sibilation if it is dental; if it escapes sibilation it is later assimilated to a preceding homorganic consonant, if any. If a weakened stop escapes all these rules it becomes *v*, *d*, or ∅ depending on whether it is labial, dental, or velar. Thus if we denote the weakened stops by the noncommittal symbols *p̆*, *t̆*, *k̆*, we have derivations something like the following:

tunten	tuntein	
tunt̆en	tunt̆ein	(gradation)
	tunt̆in	(unround-vowel deletion)
	tunsin	(sibilation)
tunnen		(assimilation)

Another problem that has concerned Anderson is the formation of the illative case of Finnish nouns and adjectives. In order to understand what this is about we must consider some further rules.

Stems ending in a consonant add a linking vowel -*e*- before most suffixes that begin with a consonant. Compare the following forms

talo	'house (nom.)'	sisar	'sister (nom.)'
talon	'house (gen.)	sisaren	'sister (gen.)'

In both cases we assume that the genitive is formed by adding *n* to the noun stem; *sisar*+*n* then becomes *sisaren* by the *e*-insertion rule.

E-insertion does not take place before certain suffixes, however; among these exceptional suffixes is the partitive *ta*. Thus we have *taloa* 'house (part.)', from *talota* by dental-stop deletion, and *sisarta* 'sister (part.)'. Whether the difference between the two types of suffixes can or ought to be described in phonological terms or in terms of diacritic features on morphemes is a question that will not concern us here. We will assume that the linking *e* appears in its proper place prior to the application of any of the rules to be discussed (in fact, prior to any of the rules discussed above).

Consider now some noun and adjective stems that end in *s*. We have *mies* 'man', *kirves* 'axe', *vieras* 'foreign', and *uros* 'male', with genitives and partitives in the singular as follows:

genitive:	miehen	kirveen	vieraan	uroon
partitive:	miestä	kirvestä	vierasta	urosta

Anderson posits three rules that will account for the above genitives. These rules, which are motivated by other evidence as well and which we see no strong reason to question, are essentially the following:

(i) intervocalic *s* after a nonfirst vowel becomes *h* ("*s*-weakening") (This rule has numerous exceptions, e.g. *kiisu* 'pyrites'),

(ii) A*he*, where A is any vowel, becomes A*h*A ("vowel-copying"),

(iii) intervocalic *h* after a noninitial [−syl] [+syl] sequence is deleted.

The derivations of the genitives are, then,

mies+n	kirves+n	uros+n	vieras+n	
miesen	kirvesen	urosen	vierasen	(*e*-insertion)
miehen	kirvehen	urohen	vierahen	(*s*-weakening)
		urohon	vierahan	(vowel-copying)
	kirveen	uroon	vieraan	(*h*-deletion)

In *miehen* the *h*, though intervocalic, is not deleted because it is preceded by a diphthong. (A more refined analysis would reveal that the underlying form of *mies* is *mees*, so that it would in fact be a long vowel that inhibits *h*-deletion.)

Let us turn now to the illative. We consider the nouns *maa* 'land', *puu* 'three', *talo* 'house', *koira* 'dog', and *katu* 'street', in the illative singular and plural.

Singular:	maahan	puuhun	taloon	koiraan	katuun
Plural:	maihin	puihin	taloihin	koiriin	katuihin

We account for all these forms with the rules we have plus one additional rule, provided we assume with Anderson that the underlying form of the illative suffix is *sen*. The additional rule, given in a somewhat different version by Anderson, deletes the middle vowel of a triphthong ("triphthong reduction"). Four derivations should make the matter clear

maasen	maaisen	koirasen	koiraisen	
			koirisen	(unround-vowel deletion)
maahen	maaihen	koirahen	koirihen	(*s*-weakening)
maahan	maaihin	koirahan	koirihin	(vowel-copying)
		koiraan	koiriin	(*h*-deletion)
	maihin			(triphthong red.)

The rules just given predict the illative form correctly when the underlying stem consists of one syllable or ends in a short vowel or a nonspirantal consonant. We must assume that illative *sen* is one of those suffixes that requires *e*-insertion; in this way we obtain the correct illative singulars *mieheen* (< *miesesen*) and *sisareen* (< *sisaresen*).

Let's turn now to the disyllabic stems ending in *s* which we mentioned above. The illative singulars *kirves*, *uros*, and *vieras* are

kirveeseen	urooseen	vieraaseen

There are two alternatives in the plural:

(a)	kirveisiin	uroisiin	vieraisiin
(b)	kirveihin	uroihin	vieraihin

There are some other disyllabic stems which in the nominative singular end in a short vowel but behave in other respects like stems ending in *s*. Anderson assumes they end in a final *h*, which is deleted when no suffix follows as well as in the environment of the *h*-deletion rule introduced previously. *h* followed by a consonant results in gemination of that consonant. Thus from the stem *kiiruh* 'hurry' we have the following case forms:

nom. sg.	kiiru
part. sg.	kiirutta
gen. sg.	kiiruun
ill. sg.	kiiruuseen
ill. pl.	kiiruisiin, kiiruihin

Polysyllabic stems ending apparently in a long vowel behave in many respects like these ending in a spirant. From the stem *tienoo* 'neighborhood' we have

nom. sg.	tienoo
part. sg.	tienoota
gen. sg.	tienoon
ill. sg.	tienooseen
ill. pl.	tienoisiin, tienoihin

The nonillative case forms of polysyllabic stems ending in a spirant or long vowel can be accounted for by the rules already introduced if appropriately ordered in the conventional manner. The illatives, with their peculiar vowel-lengthening and retention of *s*, offer some difficulty. Anderson accounts for them with a left-to-right syllabic cycle, which we give as a roughly formulated segmental cycle in (70) below.

(70) *LR*: (a) Triphthong reduction.

$$[+\text{syl}] \rightarrow \varnothing / \$^*[+\text{syl}] \,\text{—}\, [+\text{syl}][-\text{syl}]^{\circ}[+\text{syl}]\$^*$$

(b) Illative lengthening.

$$\begin{bmatrix}+\text{ill}\\ \sigma\end{bmatrix} \rightarrow \sigma\sigma / \$^* \begin{bmatrix}+\text{syl}\\ -\text{stress}\\ \tau\end{bmatrix} \tau \begin{Bmatrix} i \\ \varnothing \end{Bmatrix} [-\text{syl}]^{\circ} \,\text{—}\, \$^*$$

(c) S-weakening.

$$s \rightarrow h / \$^* \begin{bmatrix}+\text{syl}\\ -\text{stress}\end{bmatrix} \text{—}^{\circ}[+\text{syl}][-\text{syl}]\$^*$$

(d) Vowel-copying.

$$e \rightarrow \sigma / \$^* \begin{bmatrix}+\text{syl}\\ -\text{stress}\\ \sigma\end{bmatrix} h^{\circ} \,\text{—}\, \$^*$$

(e) H-vocalization.

$$h \rightarrow \sigma / \$^*[-\text{syl}] \begin{bmatrix}+\text{syl}\\ -\text{stress}\\ \sigma\end{bmatrix} \text{—}^{\circ}[+\text{syl}]\$^*$$

Of the rules in this cycle, triphthong reduction, *s*-weakening, and vowel-copying have been already introduced above, and *h*-vocalization takes the place of *h*-deletion. Illative lengthening is a new rule applying to one idiosyncratic case morpheme. Instead making use of the diacritic feature [illative], presumably introduced by a general convention that distributes syntactic class information to the individual segments of a morpheme, we could have regarded illative lengthening as a 'minor rule' from which all morphemes except the illative are exempt.

The illative singular and plural of *uros* can now be derived in part as follows:

uros$^{\circ}$esen	uros$^{\circ}$isin	
uroh$^{\circ}$esen	uroh$^{\circ}$isin	(70c)

uroh^{o}osen		(70d)
uroo^{o}osen	uroo^{o}isin	(70e)
urooo^{o}sen	urooi^{o}sin	
urooos^{o}en	uroois^{o}in	
uroos^{o}en	urois^{o}in	(70a)
uroos^{o}een		(70b)
	uroih^{o}in	(70c)

The alternative illative plural arises from a different ordering of (70a) and (70b); for example:

uros^{o}isin	
uroh^{o}isin	(70c)
uroo^{o}isin	(70e)
urooi^{o}sin	
uroois^{o}in	
uroois^{o}iin	(70b)
urois^{o}iin	(70a)

Notice that in these derivations the triphthong reduction rule comes into play only when the position marker has moved to the immediate left of the first vowel that follows the triphthong. This feature of Anderson's solution is crucial in the derivation of the illative plural form *uroisiin*, for it allows illative lengthening, formulated with ^{o} before the environment dash, to take place prior to triphthong reduction despite the fact that the illative vowel is to the right of the triphthong. Thus, when illative lengthening is ordered before triphthong reduction, *uroois^{o}in* will first become *uroois^{o}iin* and then *urois^{o}iin*. Then *s*-weakening, formulated so as to take place only before a short vowel, is blocked; thus the derivation of the nonexistent form **uroihiin* is prevented and the desired from *uroisiin* is generated.

Unfortunately, there is a serious difficulty with Anderson's formulation of the triphthong reduction rule. Consider the verb stem *saa-* 'receive'. The infinitive and first singular present are *saada* (from *saatah* by consonant gradation and final *h* deletion) and *saan*. Now the past, formed by adding *i* plus the personal suffixes, is always subject to triphthong reduction; thus we have first singular past *sain* from underlying *saain*. Similarly, the third singular past, where the personal suffix is ∅, has the form *sai*. Since no vowel occurs after the triphthong *aai* in the underlying forms of these words, Anderson's rule could not apply and *aai* would remain unreduced. Similar remarks apply to verb stems like *joo-* 'drink' and *söö-* 'eat', which have first singular presents *juon* and *syön* (from *joon* and *söön* by a vowel breaking rule) and first singular pasts *join* and *söin*, derived from *jooin* and *sööin* by triphthong reduction. A more correct and more natural formulation of triphthong reduction would seem to be

$$[+\text{syl}] \rightarrow \varnothing / \$^*[+\text{syl}] — [+\text{syl}]\$^*$$

With this formulation, of course, it is impossible to use the position marker exactly as Anderson does, although it might be possible to restore the essence of Anderson's solution by rejuggling the position marker in this and other rules. Instead of showing how this could be done, however, we will proceed to some further difficulties not as easily resolved.

The rule of *s*-weakening was originally formulated to take place regardless of whether the following vowel is long or short, but was restricted to position before short vowel in order to account for illatives. Thus *uroooseen* (or *urooseen*) and *urooisiin* are correctly prevented from undergoing *s*-weakening, as explained above. However, there seems to be no other motivation for the short vowel restriction, although indeed there seems to be no counter-evidence to it either. On the other hand, there are many apparent exceptions to *s*-weakening which are not explained by the restriction. Thus we have *kiisu* 'pyrites', *nielaisen* 'I swallow' (cf. *nielaista* 'to swallow'). Whatever the explanation of these intervocalic *s*'s may be, it has nothing to do with the length of the following vowel. This fact raises the suspicion that the simpler, less restricted *s*-weakening rule might just as well be used and that the *s* illatives are to be explained in a different way.

The third and perhaps most serious difficulty with Anderson's solution is this. Notice that Anderson's derivations of the illative plural forms of *uros* begin not with the presumed underlying form *urosisen* but with *urosisin*. Anderson does not explain the reason for this, nor does he give any rule that would change *urosisen* into *urosisin*. The change looks suspiciously like a case of vowel-copying, and can indeed be so regarded if we assume that *urosisen* first becomes *urohihen* and that the *s* is re-introduced by a revised version of the illative lengthening rule. This approach (similar to that followed by McCawley 1966) also allows us to use the simpler versions of both the triphthong reduction rule and the rule of *s*-weakening, discussed above. Furthermore, it makes it possible to apply the rules in conventional sequence. Our reformulation, still somewhat informal because not stated entirely in feature terms, is given in (71).

(71) (a) S-weakening.

$$\text{s} \rightarrow \text{h} / \$^* \begin{bmatrix} +\text{syl} \\ -\text{stress} \end{bmatrix} - [+\text{syl}]\$^*$$

(b) Vowel-copying

$$L\text{: e} \rightarrow \sigma / \$^* \begin{bmatrix} +\text{syl} \\ -\text{stress} \\ \sigma \end{bmatrix} \text{h} - \$^*$$

(c) H-vocalization.

$$R\text{: h} \rightarrow \sigma / \$^*[-\text{syl}] \begin{bmatrix} +\text{syl} \\ -\text{stress} \\ \sigma \end{bmatrix} - [+\text{syl}]\$^*$$

(d) Triphthong reduction.

$$[+\text{syl}] \rightarrow \varnothing / \$^*[+\text{syl}] - [+\text{syl}]\$^*$$

(e) Illative adjustment.

$$[+\text{ill}] \rightarrow \text{s}\sigma / \$^* \begin{bmatrix} +\text{syl} \\ -\text{stress} \\ \tau \end{bmatrix} \tau \begin{Bmatrix} \text{i} \\ \varnothing \end{Bmatrix} - \sigma\$$$

These rules apply in the order given, except that (71d) and (71e) can also be applied in the reverse order.

The derivations of the illative forms of *uros* now proceed as follows:

urosesen	urosisen	
urohehen	urohihen	(71a)
urohohen	urohihin	(71b)
uroohen	urooihin	(71c)
uroohen	uroihin	(71d)
urooseen	uroihin	(71e)

As they stand, our rules account correctly for the illatives of stems like *uros*, which end in a spirant, but not for the illatives of stems that end in a long vowel. For example, although our rules correctly predict the illative plural forms of *tienoo*, they give the wrong illative singular. By our rules, the underlying form *tienoosen* first becomes *tienoohen* by s-weakening and then *tienoohon* by vowel-copying. The illative adjustment rule then gives the incorrect result **tienoosoon*. However, we can avoid this consequence by giving *tienoo* the underlying representation *tienose* or *tienohe*. The illative singular can be then derived from underlying *tienosesen* in the same way that the illative singular of *uros* is derived from *urosesen*. If we assume that unround-vowel deletion precedes s-weakening we can derive the alternative illative plurals of *tienoo* from the underlying representation *tienoseisen* as follows. First, *tienoseisen* becomes *tienosisen*, and then *tienosisen* follows the same path as *urosisen*. The nominative singular is derived straightforwardly as follows:

tienose	
tienohe	(71a)
tienoho	(71b)
tienooo	(71c)
tienoo	(71d)

Words like *leikkuu* 'harvest' and *vapaa* 'free', which behave just like *tienoo*, can be treated similarly. They would have underlying forms like *leikkuse* or *leikkuhe* and *vapase* or *vapahe*.

An objection that might be raised against our treatment of words like *tienoo* is that a perfectly easy way of accounting for the partitive singular *tienoota* must be abandoned. If we assume with

Anderson that underlying form of *tienoo* is *tienoo*, then the retention of *t* in the partitive suffix is explained by the fact that a long vowel precedes the *t*. (cf. the environment of the cyclic dental-stop deletion rule (69a)). If we assume that the underlying representation of *tienoo* is *tienose*, then when we add the partitive suffix we have the underlying string *tienoseta*. Here the *t* is preceded by a cluster subsumed under [—syl] [+syl, —stress], which ought to trigger deletion of the *t*. The solution that immediately suggests itself is that we order dental-stop deletion after *h*-vocalization. Then we would have the derivation

tienoseta	
tienoheta	(*s*-weakening (71a))
tienohota	(vowel-copying (71b))
tienoota	(*h*-vocalization (71c))
tienoota	(triphthong reduction (71d))

(no further rules applicable)

The ordering of *h*-vocalization prior to dental-stop deletion seems to be supported independently by the illative plurals of noun and adjective stems ending in a spirant. For example, *uroita*, the illative plural of *uros*, can be derived properly from the underlying string *urosita* only if *h*-vocalization precedes dental-stop deletion.

Consider, however, the noun *airut* 'herald'. The illative plural will be underlying *airutisen*. Our rules as they now stand will convert this representation into *airuiin*. However, the correct form is *airuihin*. Observe that *airuihin* would indeed be generated if, contrary to our conclusions above, we ordered dental-stop deletion before *h*-vocalization. Then we would have the derivation

airutisen	
airutihen	(s-weakening (71a))
airutihin	(vowel-copying (71b))
airuihin	(dental-stop deletion)

h-vocalization would not apply to the last line above because the *h* is preceded by a diphthong.

The apparent conflict between forms like *uroita* and forms like *airuihin* can easily be resolved if *h*-vocalization and dental-stop deletion are regarded as subcases of a single right-linear rule. We might have thought to combine the two processes into a single rule anyway, since their environments look very similar the way we have formulated them. The similarity is not as close as it appears, however, for we have somewhat oversimplified the environment of *h*-vocalization. Apparently, *h*-vocalization takes place after any unstressed vowel sequence that does not end in a long vowel or in vowel $+i$, whereas dental-stop deletion is restricted to position after consonant + short unstressed vowel. Some effects of this difference are illustrated below.

nominative:	autio	herttua
illative:	autioon	herttuaan
partitive:	autiota	herttuata
	'desolate'	'duke'

Compare also ill. sg. *airueen* (< *airutesen*) with ill. pl. *airuihin*. The question now is how to express gracefully the environment of *h*-vocalization. I don't know a definitive answer, but perhaps we can achieve the right effect if we take the liberty of using the following somewhat controversial devices:

(i) brackets meaning set intersection, as described in Chapter 3;
(ii) hyphen meaning negation (that is, -(X) subsumes everything that X doesn't).

Then the rule of *h*-vocalization and the rule of dental-stop deletion can be combined into the single right-linear rule (72).

(72)

$$\boldsymbol{R}\colon \left\{ \begin{matrix} \begin{bmatrix} -\text{cons} \\ -\text{voice} \end{bmatrix} \rightarrow \sigma / \left[\$^* \begin{bmatrix} +\text{syl} \\ -\text{stress} \\ \sigma \end{bmatrix},\ \text{-}(\$^* \begin{bmatrix} +\text{syl} \\ \tau \end{bmatrix} \begin{Bmatrix} \text{i} \\ \tau \end{Bmatrix}) \right] \\ \begin{bmatrix} -\text{cont} \\ +\text{cor} \end{bmatrix} \rightarrow \varnothing / \$^*[-\text{syl}] \begin{bmatrix} +\text{syl} \\ -\text{stress} \end{bmatrix} \end{matrix} \right\} \text{—}[+\text{syl}]\$^*$$

It is possible that this rule can and should be further reduced by appropriate use of angle brackets or indexed braces in order to express what similarity there does exist between the left environments of the subcases.

We emphasize that our solutions to the problems under discussion are far from definitive. Numerous unresolved questions remain, but these can be answered only by a far deeper study of Finnish phonology than would be appropriate here. We could claim only that the solutions presented here are at least as plausible as Anderson's and in some respects superior, and that there is little reason to believe, therefore, that a left-to-right cycle is necessary. It is true that at least one rule, (72), must be applied in a left-to-right manner, but a right-linear formulation appears to be adequate.

FEATURES WITH INTEGRAL COEFFICIENTS

It appears from Cole's description (1955) that Tswana has the following system of underlying vowels:

i		u	High
e		o	Higher mid
ɛ		ɔ	Lower mid
	a		Low

Using the vowel features proposed in Wang (1968) we would presumably attribute the following analysis to these vowels.

	i	u	e	o	ɛ	ɔ	a
High	+	+	+	+	—	—	—
Mid	—	—	+	+	+	+	—
Palatal	+	—	+	—	+	—	—

As contrasted with the vowel features of Chomsky and Halle (1968), hereafter referred to as SPE, the above analysis has two distinct advantages in the present example: (1) it allows for the representation of four vowel heights; and (2) the vowels specified [+mid] constitute a natural class that is opposed to the remaining three vowels, as will be seen below.

In certain environments the mid vowels are raised slightly, but not enough to reach the height of the next higher vowel; in other words *ɛ* is raised but not to the extent of becoming *e*, and *e* is raised but not to the extent of becoming *i*. Phonetically, then, the following vowels are distinguished:

ɪ	u	High
e^	o^	Raised higher mid
e	o	Higher mid
ɛ^	ɔ^	Raised lower mid
ɛ	ɔ	Lower mid
a		Low

One of the raising environments is before the velar nasal ŋ. Thus we have *o*^ŋ, *e*^ŋ, *ɔ*^ŋ, and *ɛ*^ŋ from underlying *o*ŋ, *e*ŋ, *ɔ*ŋ, and *ɛ*ŋ, respectively. This rule is followed by another rule which, working from right to left, raises a mid vowel by one degree if the next following vowel is higher. We illustrate as follows:

moxakolodi	mosɛlɛsɛlɛ	mosɛlɛsɛlɛŋ	
		mosɛlɛsɛlɛ^ŋ	(raising before ŋ)
moxako^lo^di		mosɛ^lɛ^sɛ^lɛ^ŋ	(raising before higher vowel)
'advisor'	'*Dichrostachys glomerata*'	(locative of *mosɛlɛsɛlɛ*)	

The question now arises as to how we are to represent the six phonetic vowel heights in feature terms and how we are to formulate the raising rules. Clearly, the four-way height distinction that is adequate for the underlying representations is insufficiently refined for the phonetic realizations. We can, however, easily account for this situation if we allow that feature coefficients on the phonetic level be given as integers rather than as pluses and minuses. When provided with such coefficients a feature is thought of as indicating the degree to which a certain phonetic quality is possessed. Thus phonetic vowel height might be thought of as given by one of the first n positive integers, 1 indicating the least tongue height, and n indicating the greatest tongue height. The scale of height given by the integers 1 through n should presumably be universal, defined once and for all in phonological theory. In the absence of a generally accepted universal scale, we will assume ad hoc that n = 6, since this is adequate for Tswana. The rules of Tswana must at some point convert the plus and minus specifications of the high and mid features into integer specifications of the phonetic feature of height, and the raising rules will be formulated in terms of those integer specifications.

In order to handle integer specifications we will introduce a few new devices. We regard a positive integer n as a sequence of n 1's Thus 1 = 1, 2 = 11, 3 = 111, etc. The Greek letters ι, κ, λ, ... will be regarded as variables ranging over strings of 1's. We will furthermore allow expressions of the form +IF, where I is a string of 1's. Such an expression subsumes any phonological unit which contains the specification +IF or JIF where J is a string of 1's. Thus in particular, +F subsumes units containing the specifications +F, F, 1F, 11F, 111F, etc.

Suppose now that there is a rule in Tswana that converts binary height specifications into integer specifications as follows:

i,		u	become 6 high
e,		o	become 4 high
ε,		ɔ	become 2 high
	a		becomes 1 high

Then the rule that raises a mid vowel before a vowel of greater height can be given as in (75):

(75)

$$L\colon \begin{bmatrix} +\text{syl} \\ \iota 11 \text{ high} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} \iota 111 \text{ high} \\ (\sigma) \end{bmatrix} / \$^* \text{—} [-\text{syl}]^* \begin{bmatrix} +\iota 111 \text{ high} \\ +\text{syl} \end{bmatrix} \$^*$$

The vowel height feature, as we have viewed it, has finitely many possibly integer coefficients. As long as this is true of all features the alphabet of phonological units remains finite, and the extended formalism allowing integers and integer variables does not increase the mapping power of phonological rules. It is otherwise with the prosodic features now to be discussed.

According to Schachter and Fromkin (1968: 106-9) the Akan languages have an underlying distinction between high-level and low-level tone. These tones are manifested phonetically according to the following downdrift rule (after certain other rules have applied). A high tone is always on a higher pitch than an immediately neighboring low tone; however, the pitch interval between a low tone and an immediately following high tone is less than the pitch interval between a high tone and an immediately following low tone. To quantify these relations Schachter and Fromkin assume that there is a pitch interval of two degrees between a low tone and a following high tone and that there is a pitch interval of three degrees between a high tone and a following low tone. We can express this analysis in feature terms if we suppose there is an integrally valued feature of pitch, 1 pitch being the highest pitch, 2 pitch the next highest, and so on. Then the effect of the rule can be illustrated by the phrase

ɔ̀bɛ́kɔ́	kùmásé	ánɔ̀pá yí
3 1 1	4 2 2	2 5 3 3

Here high tone is indicated by an acute accent, low tone by a grave accent. High tone can be thought of as represented by the specification +tone, low tone by the specification −tone. The downdrift rule can be given by a series of three rules, of which the last is right-linear.

(76)

(a) $$\begin{bmatrix} +\text{tone} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} 1\ \text{pitch} \\ (\sigma) \end{bmatrix} /\$^* - \$^*$$

(b) $$\begin{bmatrix} +\text{syl} \\ -\text{tone} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} 111\ \text{pitch} \\ (\sigma) \end{bmatrix} /\$^* - \$^*$$

(c) $$R: \left\{ \begin{array}{l} \begin{bmatrix} +\text{tone} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} \iota\text{pitch} \\ (\sigma) \end{bmatrix} /\$^* \begin{bmatrix} +\text{syl} \\ -\text{tone} \\ \iota 11\ \text{pitch} \end{bmatrix} \left[\begin{Bmatrix} -\text{syl} \\ +\text{tone} \end{Bmatrix} \right]^* - \$^* \\ \begin{bmatrix} +\text{syl} \\ -\text{tone} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} \iota 111\ \text{pitch} \\ (\sigma) \end{bmatrix} /\$^* \begin{bmatrix} +\text{tone} \\ \iota\text{pitch} \end{bmatrix} [-\text{tone}]^* - \$^* \end{array} \right\}$$

We assume here that only syllabics can have the specification +tone, nonsyllabics being all −tone.

There are now infinitely many phonological units, since in principle [n pitch] segments exist for each n. In order to continue thinking in terms of a finite alphabet of primitive symbols we can, of course, think of a phonological string as composed of brackets, commas, pluses, minuses, 1's, and feature names, these being the formal building blocks of phonological units. We will then be in a position to consider whether the extended formalism allowing integer coefficients and variables is more powerful than the formalism of Chapters 4-5. In fact, it is more powerful, for many rules can be formulated that exceed the power of finite transducers. The Akan downdrift rule (76c), for example, is not a finite state device. To see this we can reason as follows. Let us say that Q is in the output set of a mapping device if and only if that device maps at least one string into Q. It is well known that the output set of a finite transducer is regular. Therefore, if M is some finite transducer that purportedly effects the same mapping as rule (76c), then the output set of both M and rule (76c) must be regular. However, the output set of (76c) is not regular because of the dependencies that hold between the coefficients of the pitch features within a phrase. If we regard these coefficients as strings of 1's we are obliged to say that in any output string of the form [L +tone, I pitch K −tone, J pitch H], where K contains no occurrence of the expression +tone and I and J are strings of 1's, J is longer than I by exactly two 1's. If there is no upper bound on the length of I, this sort of restriction will give rise to a non-regular set (this follows from a theorem of Nerode; see Rabin and Scott 1959 and Chomsky and Miller 1958).

Phonological rules do, then, occasionally exceed the capacity of finite-state machines. Apparently, however, this happens only in the restricted sort of case just considered, where integral coefficients are being referred to or manipulated.

(76c) has been given as right-linear but will work equally well when applied iteratively. The rule given by Schachter and Fromkin (p. 108) is in fact meant to be iterative and is essentially the same as our rule, though differing in its outward form and corresponding rather to (76) as a whole. Notice that there is no way at all to express the rule under the simultaneous mode of application. For suppose we attempted a simultaneous formulation. The determination of the pitch of each syllable would have to be made independently of the other syllabics, since the context of each syllabic must be referred to in original input form. What we would have to say, then, is this:

(i) A +tone syllabic receives n pitch when preceded in the phrase by exactly n−1 occurrences of sequences subsumed under [+tone] [−syl]* [+syl, −tone].
(ii) A −tone syllabic receives n+3 pitch when preceded in the phrase by exactly n occurrences of sequences subsumed under [+syl, −tone] [+syl]* [+tone].

However, the schematic notation as it now stands is unable to express a dependency between the number of occurrences of a particular kind of phonological string and the size of an integer coefficient, at least when there is no upper bound on these values. The most it can do in this respect is to state dependence between integers. We could, therefore, formulate a simultaneous version of (76c) only if we introduced some further device, such as the notation X^I meaning a string of exactly I X's. It would then be imperative to allow I to be a variable as well as a specific integer. However, right-linear application obviates the necessity for this extra device in the downdrift rule.

There is one other notable example of a feature that has been said to take on any positive integer as a coefficient. This feature is stress. In SPE the integer coefficients designate degrees of strength, 1 denoting the strongest stress, 2 the next strongest stress, and so on. Stress has some peculiar properties which set it apart from other features and appear to call for special conventions.

In many cases the introduction of a 1 stress causes weakening of all previously present integrally valued stresses by one degree. Thus the English alternating stress rule (rule 17, Chapter 5, in SPE) will not only place a 1 stress on the antepenultimate vowel of a final-stressed word but will also weaken the previously present final 1 stress to 2 stress. Thus *cavalc$\overset{1}{\text{a}}$de* becomes *c$\overset{1}{\text{a}}$valc$\overset{2}{\text{a}}$de*. Technically, we could treat the weakening of the final stress as a separate rule. However, weakening of this sort is such a frequent concomitant of primary stress placement that the two phenomena seem to be part of one process. Consequently Chomsky and Halle have decided in SPE to let this weakening take place automatically. If we incorporate their convention about stress into our formalism we can write the alternating stress rule simply as

$$\begin{bmatrix} +\text{syl} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} 1\ \text{stress} \\ (\sigma) \end{bmatrix} /\$^*\text{—}[-\text{syl}]^*[+\text{syl}]\ [-\text{syl}]^*[1\ \text{stress}]\ [-\text{syl}]^*$$

Many rules that introduce a primary stress operate according to a principle of disjunctive application. Consider a revised version of the alternating stress rule proposed by Ross (1969). For simplicity of illustration we take Ross' preliminary revision on page 9, omitting boundary symbols. Ross' rule can be thought of as having the two ordered subcases (77a) and (77b) (some slight modifications have been made to fit the rule to our notation).

(77)

(a) $\begin{bmatrix} +\text{syl} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} 1\ \text{stress} \\ (\sigma) \end{bmatrix} /\$^*\text{—}[-\text{syl}]^*[+\text{syl}]\ [-\text{syl}]^*\ [1\ \text{stress}][-\text{syl}]$

(b) $\begin{bmatrix} +\text{syl} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} 1\ \text{stress} \\ (\sigma) \end{bmatrix} /\$^*\text{—}[-\text{syl}]^*[1\ \text{stress}]\ [-\text{syl}]^*$

In general, all nouns are subject to (77) (most verbs are exempted from (77b) by a redundancy rule in Ross' view). However, the two cases of (77) are applied disjunctively: if a noun receives a 1 stress by (77a) it is no longer subject to (77b). Thus from *cavalc$\overset{1}{\text{a}}$de* and *kay$\overset{1}{\text{a}}$k*, where the final 1 stress has been introduced previously by the main stress rule, we obtain *c$\overset{1}{\text{a}}$valc$\overset{2}{\text{a}}$de* and *k$\overset{1}{\text{a}}$y$\overset{2}{\text{a}}$k*. It is the principle of disjunctive application that prevents the derivation

cavalc$\overset{1}{\text{a}}$de	
c$\overset{1}{\text{a}}$valc$\overset{2}{\text{a}}$de	(77a)
*c$\overset{2}{\text{a}}$v$\overset{1}{\text{a}}$lc$\overset{3}{\text{a}}$de	(77b)

It is frequently the case that the second of two disjunctively ordered rules omits material present in the first rule. This is true in (77), for example. An equivalent of (77b) can be obtained from (77a) by simply omitting the [+syl] from the expression to the right of the slash. This fact has led to the association of disjunctive ordering with parentheses and with the closely allied angle bracket notation. Thus (77) would be given in abbreviated form as (78).

(78)

$$\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}1\ \text{stress}\\ (\sigma)\end{bmatrix} / \$^* \text{—} [-\text{syl}]^*\ ([-\text{syl}])\ [-\text{syl}]^*[1\ \text{stress}][-\text{syl}]^*$$

Although parenthesis notation gives the correct result in many cases there are certain reasons why we will not incorporate it officially into our formalism. For one thing parentheses are not formally associated with the stress feature. Indeed, with their aid we can formulate many disjunctively applied rules that have nothing to do with stress. In this respect the principle of disjunctively ordering is grossly over-generalized, for it seems to be properly associated only with certain rules which introduce a primary accent (whether of pitch or stress). Supposed cases of disjunctive ordering that do not fall under this rubric appear to be spurious. A few illustrative examples will suggest why I believe this to be so.[1]

[1] Chomsky and Halle (p. 366 of SPE) are aware of the sort of problem about to be discussed and offer some preliminary suggestions for solving it. We have not followed up these suggestions because the problem does not arise within the rule formalism we are developing here. It must be granted that the case discussed by Chomsky and Halle, a rule of Latvian, presents a difficulty of another kind to our formalism. Adapted slightly to our notation, but retaining the parentheses and morpheme boundary of Chomsky and Halle's formulation, the rule would appear as follows:

$$\begin{bmatrix}+\text{syl}\\ +\text{high}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}-\text{syl}\\ (\sigma)\end{bmatrix} / \ \$^* \text{ — } (+)\ [+\text{syl}]\ \$^*$$

This is taken to represent the following sequence of rules:

$$\text{(a)}\quad \begin{bmatrix}+\text{syl}\\ +\text{high}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}-\text{syl}\\ (\sigma)\end{bmatrix} / \ \$^* \text{ — } +\ [+\text{syl}]\ \$^*$$

One of the rules proposed in SPE (23III in Chapter 5) replaces lax *u* with tense ɨ̄ before a string subsumed under [−syl][−cons] or [−cons]. This appears to cry out for parenthesis notation, and the authors of SPE do in fact use an equivalent notation involving a subscript 0 and a superscript 1. Adapted slightly to fit our formalism more closely, the rule could be given as (79).

$$(79)\quad \begin{bmatrix} +\text{syl} \\ -\text{tns} \\ +\text{back} \\ +\text{high} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{tns} \\ -\text{rnd} \\ (\sigma) \end{bmatrix} / \$^* - [-\text{syl}]_0^1 [-\text{cons}]\$^*$$

According to the conventions in SPE (pp. 61-2, 199), (79) would be expanded into the two ordered rules of (80),

$$(80)\quad \begin{array}{l} \text{(a)}\ \text{u} \rightarrow \bar{\text{ɨ}} / \$^* - [-\text{syl}][-\text{cons}]\$^* \\ \text{(b)}\ \text{u} \rightarrow \bar{\text{ɨ}} / \$^* - [-\text{cons}]\$^* \end{array}$$

Now consider the effect of the u → ɨ̄ rule on the word *usual*. This word presumably has the underlying phonological representation *usuæl* (SPE, p. 228). S-voicing apparently should apply before the u → ɨ̄ rule since s-voicing precedes velar softening according to SPE, p. 221, and velar softening precedes the u → ɨ̄ rule 23III of Chapter 5. The input to the u → ɨ̄ rule will, then, be *úzuæl*. This string has two vowels which fit the conditions of the u → ɨ̄ rule, the first *u* satisfying case (80a) and the second *u* satisfying case (80b). According to the conventions of SPE, however, the two cases are applied disjunctively; consequently only the first case (80a) will apply, and the output will be *ɨ́zuæl*. Unfortunately,

$$\text{(b)}\quad \begin{bmatrix} +\text{syl} \\ +\text{high} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} -\text{syl} \\ (\sigma) \end{bmatrix} / \$^* - [+\text{syl}]\ \$^*$$

When applied in the indicated order (a) and (b) will correctly convert *kuru+iai* and *aui+a* into *kurw+yai* and *auy+a*, respectively. As our formalism now stands we cannot, as Chomsky and Halle can, regard (a) and (b) as subcases of a single rule. If they are subcases of a single right-linear or simultaneous rule they yield **awy+a* instead of *auy+a*; if they are subcases of a single left-linear rule, they yield **kuru+yai* instead of *kurw+yai*.

though, this output cannot yield the correct pronunciation of the word, for the remainder of the derivation would be as follows (numbers not in parentheses refer to rules in Chapter 5 of SPE):

ɨ́zuæl	
ɨ́zūæl	23IV
yɨ́zūæl	29
yɨ́wzūwæl	31
yūwzūwæl	34
*yūwzūwəl	43

The desired phonetic form is *yū́wžūwəl*, with palatalization of the *z*, But it is just this correct form that will be obtained of the two cases (80a) and (80b) of the u → ɨ̄ rule are applied conjunctively or simultaneously. Then the input string *úzuæl* will be converted to *ɨ́zɨ̄æl* and the remainder of the derivation will proceed as desired:

ɨ́zɨ̄æl	
yɨ́zyɨ̄æl	29
yɨ́wzyɨ̄wæl	31
yū́wzyūwæl	34
yū́wžyūwæl	37
yū́wžūwæl	38
yū́wžūwæl	43

Note that within our formalism, where there is no explicit analogue to the notion of ordered expansion of rules, (79) will be interpreted correctly if the notation $[-\text{syl}]_0^1$ is regarded as equivalent to $\{[-\text{syl}], \varnothing\}$ and if (79) is regarded as a simultaneous, right-linear, or left-linear rule. For then (79) will convert *úzuæl* to *ɨ́zɨ̄æl* by virtue of the application

$$\varnothing \; \begin{matrix}\text{ú}\\ \text{ɨ́}\end{matrix} \; \text{z} \; \begin{matrix}\text{u}\\ \text{ɨ̄}\end{matrix} \; \text{æl}$$

Variable notation too has been interpreted according to the principle of disjunctive ordering, with equally disastrous consequences. Consider the second part of the vowel shift (rule 33 of Chapter 5 in SPE), which could be given as

$$(81)\quad \begin{bmatrix} \text{X} \\ \beta\text{low} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} -\beta\text{low} \\ (\sigma) \end{bmatrix} /\$^* - \$^*,$$

where X is the expression

+syl
+tns
+stress
—high
γback
γround

According to the convention described on p. 357 of SPE, (81) must be regarded as standing for the disjunctive sequence (82).

$$(82)\quad (a)\ \begin{bmatrix} \text{X} \\ +\text{low} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} -\text{low} \\ (\sigma) \end{bmatrix} /\$^* - \$^*$$

$$(b)\ \begin{bmatrix} \text{X} \\ -\text{low} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} +\text{low} \\ (\sigma) \end{bmatrix} /\$^* - \$^*$$

Now consider the effect on the word *migrate*. The underlying representation is presumably *mīgrǣt* (in SPE, p. 144, this word is thought to have a + boundary before the *æ*, but this fact is of no relevance here except for its effect on the stress). The stress rules, diphthongization and the first part of the vowel shift rule will convert *mīgrǣt* into the intermediate form *mḗygrǽ̀yt*. Applying case (82a) of the second part of the vowel shift we obtain *mḗygrè̄yt*. Now, however, because of disjunctive ordering, (82b) cannot now apply to this output. But this is not what is wanted; the desired output is *mǣ́ygrè̄yt*. Conjunctive ordering of (82a) and (82b) will not do either, for then the output *mḗygrè̄yt* will be obtained. Within the framework of SPE the most appropriate method of application seems to be the simultaneous; for then (82a) will apply to the second vowel but not to the first, and (82b) will apply to the first vowel but not to the second, as required. Within our formalism the correct result is obtained immediately if (81) is regarded as simultaneous, right-linear, or left-linear. Then the rule will convert *mḗygrǣ̀yt* into *mǽygrèyt* by virtue of the application

$$\text{m}\begin{array}{c}\text{é}\\ \text{ǽ}\end{array}\text{ygr}\begin{array}{c}\text{æ̀}\\ \text{è}\end{array}\text{t}$$

There are several more cases in SPE where the notation incorrectly calls for disjunctive ordering; the amusing game of finding them is left to the reader. There are, in addition, a number of cases where disjunctive application is called for by the notation but is neither supported nor invalidated by any crucial evidence in SPE. Rule 24 of Chapter 5 in SPE is a particularly interesting example. The first part of the rule would take the following form within our notation:

(83) $$Y(Z)X\begin{Bmatrix}\emptyset\\ W\end{Bmatrix} : K$$

where

$$Y = \begin{bmatrix}\alpha\text{stress}\\ +\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}11\ \text{stress}\\ (\sigma)\end{bmatrix} / [-\text{stress}]^* \,—\, [-\text{syl}]^*,$$

$$Z = \begin{bmatrix}+\text{syl}\\ -\text{tns}\\ -\text{stress}\end{bmatrix}([-\text{syl}]),$$

$$X = [-\text{syl}]^*\begin{bmatrix}+\text{syl}\\ \beta\text{stress}\end{bmatrix}[-\text{syl}]^*\begin{bmatrix}+\text{syl}\\ \gamma\text{stress}\end{bmatrix}[-\text{syl}]^*,$$

$$W = \begin{bmatrix}\delta\text{stress}\\ +\text{syl}\end{bmatrix}\$^*,$$

and K is the condition

$(-(\alpha = 1)$ and $-(\beta = 1)$ and $-(\beta = 11)$ and $-(\gamma = \iota\delta))$ (that is, α is not 1, β is weaker than 2, and δ is weaker than γ).

According to the conventions of SPE, though not according to our conventions, (83) stands for the following sequence:

```
(84)   (a)  (i)  YZX:K
            (ii) YX:K
       (b)  (i)  YZXW:K
            (ii) YXW:K
```

(84a) and (84b) are conjunctively ordered. Within each of (84a) and (84b) the two subcases (i) and (ii) are said by Chomsky and Halle to be disjunctively ordered (see Footnote 67, p. 114 of SPE, where a preliminary formulation of the rule is discussed). Contrary to what they imply, however, it is far from evident that only disjunctive ordering would yield the correct results. One can show formally that (84a) can be applied either disjunctively or conjunctively with no difference in effect on any logically possible input. The following derivation illustrates what happens:

```
rɔdAmAntǣd   (rodomontade; A is some lax vowel)
        1    (main stress rule; the word is apparently an excep-
             tion to the alternating stress rule)
      2      (84a.i)
```

Even if (84a) is a conjunctive sequence, case (ii) cannot now apply because the word cannot be matched to the structural description in any way. In particular, the second vowel is not subject to the rule because it is preceded in the word by a stressed segment.

(84b) will indeed treat certain inputs differently depending on the way it is applied. A hypothetical example will illustrate how this might happen:

```
winAtætAmAgōči   (?Winnetatamagouchi; A is some lax vowel)
            1    (main stress rule)
     2           (84b.i)
 2               (84b.ii)
```

Both of the last two steps are taken if (84b) applies conjunctively, but the third step is omitted if (84b) applies disjunctively. To decide whether (84b) should be conjunctive or disjunctive one would have to consider some long English word whose canonical form was similar to that of our hypothetical example. In discussing the rule (SPE, pp. 114-15), Chomsky and Halle consider only shorter words that prove nothing either way. One of these words is *Winnepesaukee*, whose stress pattern is derived as follows:

```
winApAsāki      (underlying form; A a lax vowel)
       1        (main stress rule)
 2              (84b.i)
```

Even if (84.b) is a conjunctive sequence, case (ii) is now inapplicable because of its structural description. Thus, in particular, the second vowel of the word cannot be touched because it is now preceded somewhere by a stressed segment.

It would seem, then, that disjunctive ordering is a principle of highly limited scope, being clearly required only by certain rules of primary accentuation. Furthermore, if we assume that all stress rules introduce either a −stress or an integer-valued stress, but never a +stress as such, then even many rules of primary stress placement cannot be applied disjunctively. For example, if we recast the Southern Paiute rule (47) so as to introduce a 1 stress instead of a +stress, we want to obtain $tuk^{w}\overset{1}{a}pa\overset{1}{i}yu$, not $*tuk^{w}\overset{1}{a}paiyu$. It may be a further fact that disjunctive application of a primary stress rule and weakening of previously present stresses always go together; at least this is true of the English stress rules. These considerations lead us to propose the conventions of (85).

(85) (a) The stress feature may have the coefficient 0 as well as − and positive integers. However, no underlying representation may contain a 0 stress vowel.

(b) The definition of subrule is revised as follows. P → Q/R—S is a subrule of G:X if and only if P → Q/R—S is subsumed under X and RPS contains no 0 stress vowel. Q, however, may contain such a vowel.

(c) Let Q be the result of applying a rule N to the input string P. If Q contains at least one 0 stress vowel, then all integrally valued stresses in Q, including the 0 stresses, are weakened by one.

With these conventions we can now formulate Ross' version of the alternating stress rule as follows:

$$\text{(86)}\quad R\!: \begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}0\ \text{stress}\\ (\sigma)\end{bmatrix} /\$^{*}\text{—}[-\text{syl}]^{*}\begin{Bmatrix}[+\text{syl}]\\ \varnothing\end{Bmatrix}[-\text{syl}]^{*}$$
$$[1\ \text{stress}]\,[-\text{syl}]^{*}$$

Application of (86) to $\textit{kævælk}\overset{1}{\bar{\textit{æ}}}\textit{d}$ (perhaps the representation of *cavalcade* at this stage of derivation) will take the following form:

$$\text{k}\begin{matrix}\text{æ}\\ \overset{0}{\text{æ}}\end{matrix}\ \text{vælk}\overset{1}{\bar{\text{æ}}}\text{d}$$

The output $\textit{k}\overset{0}{\textit{æ}}\textit{vælk}\overset{1}{\bar{\textit{æ}}}\textit{d}$ is turned into $\textit{k}\overset{1}{\textit{æ}}\textit{vælk}\overset{2}{\bar{\textit{æ}}}\textit{d}$ by convention (85c). Notice that (85) cannot give the output $\textit{k}\overset{0}{\textit{æ}}\textit{v}\overset{0}{\textit{æ}}\textit{lk}\overset{1}{\bar{\textit{æ}}}\textit{d}$ by virtue of any putative application

$$\text{k}\begin{matrix}\text{æ}\\ \overset{0}{\text{æ}}\end{matrix}\ \text{v}\begin{matrix}\text{æ}\\ \overset{0}{\text{æ}}\end{matrix}\ \text{lk}\overset{1}{\bar{\text{æ}}}\text{d}$$

The reason is that the supposed fourth step,

$$\text{æ} \rightarrow \overset{0}{\text{æ}}/\text{k}\overset{0}{\text{æ}}\text{v} \text{ — } \text{lk}\overset{1}{\bar{\text{æ}}}\text{d},$$

though subsumed under the schema in (86), is not a subrule of (85), being excluded by convention (85b).

Ross' version of the main stress rule of English, though not the version in SPE, fits quite well into the framework proposed here. Adapted slightly to our notation, Ross' rule as given on p. 45 takes the form of (87).

(87)
$$\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}1\ \text{stress}\\ (\sigma)\end{bmatrix} /\$^{*}\text{—}[-\text{syl}]^{*}\left(\left(\begin{bmatrix}+\text{syl}\\ -\text{tns}\end{bmatrix}([-\text{syl}])\right)\begin{bmatrix}+\text{syl}\\ -\text{tns}\end{bmatrix}\text{K}\right)\text{L}$$

where K is the expression

$$\left(\left\{\left\langle\left\langle\begin{matrix}[+\text{son}]\\ \text{s}\\ (\text{n})\,\text{t}\\ \begin{bmatrix}+\text{cor}\\ +\text{ant}\end{bmatrix}\\ \left\{\begin{matrix}\text{s}\\ \text{r}\end{matrix}\right\}\text{t}\\ \text{r}\left\{\begin{matrix}\text{d}\\ \text{n}\end{matrix}\right\}\\ \left\{\begin{matrix}\text{n}\\ \text{t}\end{matrix}\right\}\text{s}\end{matrix}\right\rangle_a\right\rangle_b\right\}\right)$$

and L is the right syntactic bracket

$$\Big]_{{}_b\langle{}_a\langle N\rangle_a A\rangle_b}$$

According to the conventions of SPE (cf. especially the Appendix to Chapter 8), rule (87) will have the partial expansion shown in (88).

(88)

a) $\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}1\ \text{stress}\\ (\sigma)\end{bmatrix} / \$^* \text{—} [-\text{syl}]^* \begin{bmatrix}+\text{syl}\\ -\text{tns}\end{bmatrix} ([-\text{syl}]) \begin{bmatrix}+\text{syl}\\ -\text{tns}\end{bmatrix} KL$

(b) $\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}1\ \text{stress}\\ (\sigma)\end{bmatrix} / \$^* \text{—} [-\text{syl}]^* \begin{bmatrix}+\text{syl}\\ -\text{tns}\end{bmatrix} KL$

(c) $\begin{bmatrix}+\text{syl}\\ \sigma\end{bmatrix} \rightarrow \begin{bmatrix}1\ \text{stress}\\ (\sigma)\end{bmatrix} / \$^* \text{—} [-\text{syl}]^* L$

The three cases (88a), (88b), and (88c) of rule (87) stress antepenultimate, penultimate, and final vowels, respectively. Since the cases are applied disjunctively in the order given, the effect is to stress the leftmsot vowel that fits any of the three cases. We can, then, adapt (87) completely to our formalism with very minor revisions, provided we adopt the customary angle bracket notation and the usual way of referring to syntactic categories. All we need to do is to make (87) a right-linear rule, substitute the expression [0 stress, (σ)] for [1 stress, (σ)] to the right of the arrow, and replace each expression of the form (U) with {U, ∅}.

There are a few cases where disjunctive application seems appropriate and is in fact achievable under our conventions but not under those of SPE. According to Harms (1968: 74), Komi Jaźva has a word-level rule that places stress on the rightmost vowel that is not preceded anywhere within the word by a tense vowel. Thus if there is a tense vowel, the first tense vowel receives stress, while if there is no tense vowel, then the last vowel receives stress. With our conventions we would write the rule as follows:

(89) $$L\colon \begin{bmatrix} +\text{syl} \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} 0\ \text{stress} \\ (\sigma) \end{bmatrix} / \left[\begin{Bmatrix} -\text{syl} \\ -\text{tns} \end{Bmatrix}\right] {*}\_\_ \$^{*}$$

There is no way in the system of SPE to achieve the effect of disjunctive application here, since in SPE the star notation is associated with simultaneous application. Parentheses around the left-hand starred expression will not do either, since the first of the two disjunctively ordered cases will be (89) itself.

REFERENCES

Allen, W.S. 1951. "Some prosodic aspects of retroflexion and aspiration in Sanskrit", *BSOAS* 13: 939-46.

Anderson, L.B. 1968. "The left-to-right syllabic cycle", PEGS Paper No. 32.

Bendor-Samuel, J.T. 1960. "Some problems of segmentation in the phonological analysis of Terena", *Word* 16: 348-55.

Bennett, P. 1967. "Dahl's Law and Thagicũ", *African Language Studies* 8: 127-59.

Bloomfield, L. 1956. *Eastern Ojibwa* (Ann Arbor, University of Michigan Press).

Bobrow, D.G., and J.B. Fraser. 1968. "A phonological rule tester", *Communications of the ACM* 11: 766-72.

Burssens, A. 1946. *Manuel de Tshiluba* (Antwerp, De Sikkel).

Chomsky, N. 1963. "Formal properties of grammars", in *Handbook of Mathematical Psychology* (Luce, R.D., R.S., Bush, and E. Galanter, eds.) Vol. II (New York, John Wiley), 323-418.

Chomsky, N., and M. Halle. 1968. *The Sound Pattern of English* (New York, Harper and Row).

Chomsky, N., and G. Miller. 1958. "Finite state languages", *Information and Control* 1: 91-112.

Cole, D.T. 1955. *Introduction to Tswana Grammar* (London, Longman's).

Coupez, A. 1954. *Études sur la langue Luba* (= *Annales de Musée Royal du Congo Belge. Série in 8°. Sciences de l'homme: Linguistique* No. 9) (Tervuren, Belgium).

Davis, M. 1958. *Computability and Unsolvability* (New York, McGraw-Hill).

Delattre, P. 1951. "Le jeu de l'*e* instable intérieur en français", *French Review* 24: 341-51.

Emeneau, M.B., and B. van Nooten, B.A. 1968. *Sanskrit Sandhi and Exercises* (Berkeley and Los Angeles, University of California Press).

Fromm, H., and M. Sadeniemi. 1956. *Finnisches Elementarbuch* (Heidelberg, Carl Winter).

Ginsburg, S. 1962. *An Introduction to Mathematical Machine Theory* (Reading, Mass., Addison-Wesley).

Ginsburg, S., and B. Partee. 1969. "A mathematical model of transformational grammars", *Information and Control* 5: 297-334.

Han, M.S. 1962. *Japanese Phonology* (Tokyo, Kenkyusha).

Harms, R.T. 1964. *Finnish Structural Sketch* (= *Indiana University Publications, Uralic and Altaic series* 42) (Bloomington, Indiana University).

——, 1966a. "The measurement of phonological economy", *Language* 42: 602-11.

——, 1966b. "Stress, voice, and length in Southern Paiute", *IJAL* 32: 228-35.

——, 1968. *Introduction to Phonological Theory* (Englewood Cliffs, N.J., Prentice-Hall).

Havlík, A. 1889. "K otázce jerové v staré češtině", *Listy Filologické* 16.

Hetzron, R. 1967. "Agaw numerals and incongruence in Semitic", *Journal of Semitic Studies* 12: 169-97.

——, 1969. The verbal system of Southern Agaw (= *University of California Publications, Near Eastern Studies* 12) (Berkeley and Los Angeles, University of California Press).

Kimball, J.P. 1967. Predicates definable over transformational derivations by intersection with regular languages", *Information and Control* 11: 177-95.

Kiparsky, P. 1968. "Linguistic universals and linguistic change", *Universals in Linguistic Theory* (Bach, E., and R.T. Harms, eds.), 171-202.

Kisseberth, C.W. 1969. "On the abstractness of phonology: the evidence from Yawelmani", *Papers in Linguistics* 1: 248-82.

Kuroda, S-Y. 1967. *Yawelmani Phonology* (Cambridge, Mass., MIT Press).

Langacker, R.W. 1969. "Mirror image rules II: lexicon and phonology", *Language* 45: 844-62.

Langendoen, D.T. 1968. *The London School of Linguistics* (Cambridge, Mass., MIT Press).

Lehtinen, M. 1967. *Basic Course in Finnish* (= *Indiana University Publications, Uralic and Altaic Series* 27) (Bloomington, Indiana University Press).

Li, F.-K. 1946. Chipewyan. *Linguistic Structures of Native America* (C. Osgood, ed.) (New York, Viking Fund), 398-423.

McCawley, J.D. 1963. "Finnish noun morphonology", MIT Research Laboratory of Electronics, *Quarterly Progress Reports* 68: 180-86.

——, 1966. "Further revisions of Finnish rules". Unpublished.

——, 1968. *The Phonological Component of a Grammar Japanese* (The Hague, Mouton).

——, 1969. "Length and voicing in Tübatulabal", *Papers from the Fifth Regional Meeting of the Chicago Linguistic Society*, 407-15.

McNaughton, R., and H. Yamada. 1960. "Regular expressions and state graphs for automata", *Transactions of the IRE Professional Group on Electronic Computers*, EC-9, 39-47.

Peters, S., and R. Ritchie. 1969. "A note on the universal base hypothesis", *Journal of Linguistics* 5: 151-52.

Postal, P.M. 1969. "Mohawk vowel doubling", *IJAL* 35: 291-98.

Rabin, M.O., and D. Scott. 1959. "Finite automata and their decision problems", *IBM Journal of Research and Development* 3: 114-25.

Romeo, L. 1964. "Toward a phonological grammar of Modern Spoken Greek", *Word*, Special publication 5: 60-78.

Ross, J.R. 1969. "A reanalysis of English word stress." Preliminary unpublished version.

Sapir, E. 1930. "The Southern Paiute Language", *Proceedings of the American Academy of Arts and Sciences* 65, Nos. 1, 2, and 3.

Schachter, P., and V. Fromkin. 1968. *A Phonology of Akan* (= *Working Papers in Phonetics* 9) (Los Angeles, University of California).

Schützenberger, M.P. 1961. "A remark on finite transducers", *Information and Control* 4: 185-96.

Shevelov, G.Y. 1965. *A Prehistory of Slavic* (New York, Columbia University Press).

Swadesh, M., and C.F. Voegelin. 1939. "A problem in phonological alternation", *Language* 15: 1-10.

Voegelin, C.F. 1935. "Tübatulabal grammar", *University of California Publications in American Archaeology and Ethnology* 34: 55-190.

Wang, W. S-Y. 1968. "Vowel features, paired variables, and the English vowel shift", *Language* 44: 695-708.

Warotamasikkhadit, U. 1964. "Phonotactics, Markov chains, and transformations", *General Linguistics* 6: 21-23.

Whitney, W.D. 1889. *Sanskrit Grammar* (Cambridge, Mass., Harvard University Press).

Zwicky, A.M. 1965. "Selected topics in Sanskrit phonology". MIT unpublished doctoral dissertation.