One-Level Phonology: Autosegmental Representations and Rules as Finite Automata

Steven Bird*
University of Edinburgh

T. Mark Ellison*
University of Edinburgh

When phonological rules are regarded as declarative descriptions, it is possible to construct a model of phonology in which rules and representations are no longer distinguished and such procedural devices as rule-ordering are absent. In this paper we present a finite-state model of phonology in which automata are the descriptions and tapes (or strings) are the objects being described. This provides the formal semantics for an autosegmental phonology without structure-changing rules. Logical operations on the phonological domain—such as conjunction, disjunction, and negation—make sense since the phonological domain consists of descriptions rather than objects. These operations as applied to automata are the straightforward operations of intersection, union, and complement. If the arrow in a rewrite rule is viewed as logical implication, then a phonological rule can also be represented as an automaton, albeit a less restrictive automaton than would be required for a lexical representation. The model is then compared with the transducer models for autosegmental phonology of Kay (1987), Kornai (1991), and Wiebe (1992). We conclude that the declarative approach to phonology presents an attractive way of extending finite-state techniques to autosegmental phonology while remaining within the confines of regular grammar.

1. Introduction

The decade since the publication of Koskenniemi's dissertation (1983) and since the development of the KIMMO system (Karttunen 1983) has witnessed a spectacular flurry of activity as the linguistic and computational consequences of this work have been fleshed out. A considerable body of literature has grown up around TWO-LEVEL MORPHOLOGY, along with texts[1] and implementations.[2] The existence of a rule compiler (Koskenniemi 1985) has made it possible for the linguist to work at a conveniently abstract level, and analyses of several languages now exemplify the approach. Today, two-level morphology encompasses much of traditional segmental generative phonology of the SPE variety (Chomsky and Halle 1968).[3]

Although further development and application of this model is set to continue for some time, there is now a clear need to integrate it more closely with computational

* University of Edinburgh, Centre for Cognitive Science, 2 Buccleuch Place, Edinburgh EH8 9LW, Scotland, U.K. E-mail: {steven,marke}@cogsci.ed.ac.uk

1 (Antworth 1990; Ritchie, Russell, Black, and Pulman 1992; Sproat 1992)

2 (Bear 1986; Antworth 1990; Schiller and Steffens 1991; Pulman and Hepple 1993)

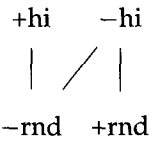
3 Two caveats are necessary here. SPE rules must be restricted so as not to apply to their own output (Johnson 1972) and there is no guarantee that the transducer encoding an SPE rule can be expressed using the two-level rule notation (Ritchie 1992).

grammar frameworks on the one hand and modern nonlinear phonology on the other. The primary goal of this article is to show how the central tenets of autosegmental phonology translate into an implemented finite state model.

The model is named ONE-LEVEL PHONOLOGY for two reasons. First, the model is monostratal, in that there is only *one level* of linguistic description. Second, the name is intended to contrast with models employing two levels (such as the FST model mentioned above) or three levels (Goldsmith 1991; Touretzky and Wheeler 1990), or an unbounded number of levels (Chomsky and Halle 1968). The one-level model represents the outgrowth of three independent strands of research: (i) the finite-state modeling of phonology, (ii) the declarative approach to phonology,[4] and (iii) the automatic learning of phonological generalizations (Ellison 1992, 1993).

The paper is organized as follows. Section 2 presents an overview of autosegmental phonology and the temporal semantics of Bird and Klein (1990). Then we define state-labeled automata (Section 3.1), show their equivalence to finite state automata (Section 3.2), define the operations of concatenation, union, intersection, and complement (Section 3.3), and further define state-labeled transducers (Section 3.4). The central proposals of the paper are contained in Section 4. We show how autosegmental association can be interpreted in terms of the synchronization of two automata, where each automaton specifies an autosegmental tier. We now give a brief foretaste of this procedure. Suppose that we have the autosegmental diagram in (1), encoding high (hi) and round (rnd) autosegments.

1.



This diagram is encoded as the following expression, where each numeral indicates the number of association lines incident with its corresponding autosegment.

$$\begin{array}{cc} +\text{hi}{:}1 & -\text{hi}{:}2 \\ -\text{rnd}{:}2 & +\text{rnd}{:}1 \end{array}$$

From this encoding, we can write down the following regular expression. Although such expressions will be opaque at this early stage of the exposition, it suffices to note here that each line of the expression represents a tier and the tiers are combined using the intersection operation ($\sqcap$). Moreover, the 1s act as synchronization marks between the operands of the intersection operation.

$$\begin{array}{ll} & \langle +\text{hi}, 0\rangle^* \langle +\text{hi}, 1\rangle \langle +\text{hi}, 0\rangle^* \langle -\text{hi}, 0\rangle^* \langle -\text{hi}, 1\rangle \langle -\text{hi}, 0\rangle^* \langle -\text{hi}, 1\rangle \langle -\text{hi}, 0\rangle^* \\ \sqcap & \langle -\text{rnd}, 0\rangle^* \langle -\text{rnd}, 1\rangle \langle -\text{rnd}, 0\rangle^* \langle -\text{rnd}, 1\rangle \langle -\text{rnd}, 0\rangle^* \langle +\text{rnd}, 0\rangle^* \langle +\text{rnd}, 1\rangle \langle +\text{rnd}, 0\rangle^* \end{array}$$

The final step is to compute the intersection and project the first element of each tuple (ignoring the 1s and 0s). This produces the expression:

$$(+\text{hi} \cap -\text{rnd})^+ (-\text{hi} \cap -\text{rnd})^+ (-\text{hi} \cap +\text{rnd})^+.$$

4 Wheeler 1981; Bird 1990; Coleman 1991; Scobbie 1991; Broe 1993; Russell 1993; Mastroianni 1993.

Given plausible interpretations of the high and round features, this last expression simplifies to $i^+a^+o^+$, which describes an automaton tape (or a string) divided into three nonempty intervals, the first containing [i], the second containing [a], and the third containing [o]. This, we shall claim, is the intended interpretation of (1).

After a detailed discussion of this procedure, the remainder of Section 4 is given over to generalizing the procedure to an arbitrary number of autosegmental charts (Section 4.4), an evaluation of the encoding with respect to Kornai's desiderata (Section 4.5), and a presentation of the encoding of autosegmental rules (Section 4.6).

Finally, Section 5 compares our proposals with those of Kay (1987), Kornai (1991), and Wiebe (1992). While our model has regular grammar power and is fully implemented, these three models go beyond regular grammar power and to our knowledge have never been implemented.

2. Background

It has long been recognized that the SPE model lacks explanatory adequacy, a fact noted in SPE itself (Chomsky and Halle 1968, pp. 400ff). For example, it is unable to explain why a final devoicing rule like that in (2a) is commonplace in the languages of the world, whereas the rule in (2b) is unattested (Kaye 1989, p. 61).

2. (a) [−sonorant] → [−voice] / — #
   (b) [−sonorant] → [+nasal] / — #
   (c) [] → [αnasal] / [αnasal] — #
   (d) [] → [αround] / [αnasal] — #

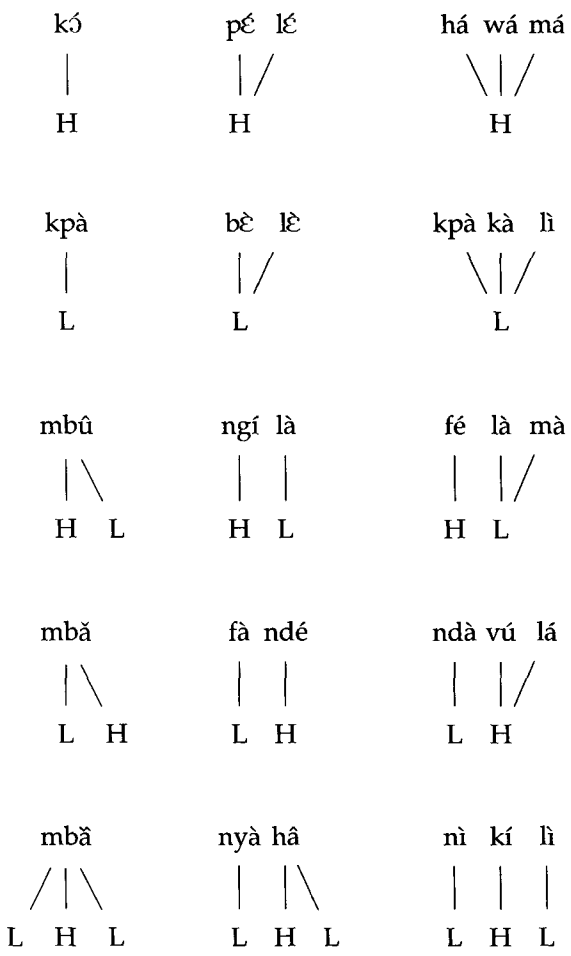
Similarly, the nasal harmony rule in (2c) occurs frequently, while the generalization expressed in (2d) is as unlikely as (2b). Both SPE and the two-level model are unable to express the fact that some rules are commonplace while others are highly unnatural.

Perhaps the SPE model could be rescued from these problems with additional stipulations. However, a more fundamental problem for the model was raised by the following tone language data (Leben 1973; Goldsmith 1976). At first blush, Mende vowels appear to manifest five tone patterns, namely high (kɔ), low (kpà), falling (mbû), rising (mbă) and rise-fall (mbă). The SPE model would predict 25 tonal patterns for two-syllable morphemes, but instead we find only 5. These are high-high, low-low, high-low, low-high, and low-falling. Similarly, for three-syllable morphemes we get 5 patterns, not 125. Leben noticed that the one-, two-, and three-syllable morphemes could be put into correspondence as shown in (3), from Leben (1978).

3.							
	H:	kɔ́	*war*	pɛ́lɛ́	*house*	háwámá	*waistline*
	L:	kpà	*debt*	bɛ̀lɛ̀	*trousers*	kpàkàlì	*tripod chair*
	HL:	mbû	*owl*	ngílà	*dog*	félàmà	*junction*
	LH:	mbă	*rice*	fàndé	*cotton*	ndàvúlá	*sling*
	LHL:	mbâ	*companion*	nyàhâ	*woman*	nìkílì	*groundnut*

Goldsmith (1976) devised a graphical notation that made the above correspondence clearer still. We display several examples of his notation in (4). Here, the **H** indicates high tone, while **L** indicates low tone. (The association lines are assumed to be incident with the vowels.)

4.



Observe that each row of the table has the same tone pattern. Only the synchronization varies. In diagrams like the ones in (4), units such as H and L are termed AUTONOMOUS SEGMENTS, or AUTOSEGMENTS, and a linear sequence of autosegments is called a TIER. The synchronization markers are called ASSOCIATION LINES. A pair of tiers linked by some association lines is called a CHART. A chart is called WELL-FORMED if the following conditions hold (Goldsmith 1976, p. 27).

5. **Well-Formedness Condition:**

(a) All vowels are associated with at least one tone; all tones are associated with at least one vowel.
(b) Association lines do not cross.

The reader can ascertain that the above charts are well-formed according to (5). However, (5) is insufficiently restrictive on its own, and a further stipulation is required.

6. **Association Convention:** Only the rightmost member of a tier can be associated to more than one member of another tier.

When (5) and (6) are combined, we achieve the effect of one-to-one left-to-right association, where multiple association (or SPREADING) occurs only at the right-hand end. Observe also that the charts in (4) do not contain adjacent identical tones. For example, there is no HH tone melody. This is expressed by a principle attributable to Leben (1973).

7. **Obligatory Contour Principle:** At the melodic level of the grammar, any two adjacent [autosegments] must be distinct. Thus HHL is not a possible melodic pattern; it automatically simplifies to HL.

Although nonlinear models like autosegmental phonology represent a major advance on the linear model of SPE in the area of explanatory adequacy, it has sometimes been pointed out (e.g., Bird and Ladd 1991) that the formal explicitness of the SPE model has not been matched by these more recent proposals. Before we can begin to compute with autosegmental representations and rules, they need to be given a formal semantics. Our starting point here is the temporal semantics of Bird and Klein (1990), based on Sagey's (1988) model, which has gained widespread acceptance in autosegmental phonology. Under this temporal semantics, phonological properties are attached to intervals that are related using precedence (an asymmetric, transitive relation) and overlap (a reflexive, symmetric relation).

Bird (1990) showed how a phonological description language can be modeled by such event structures, where the precedence relation models the linear ordering of tiers and the overlap relation models association lines. In this paper, we shall provide an automaton-based semantics for precedence and overlap, thus arriving at a computational semantics for the autosegmental notation.

3. State-Labeled Automata

In this section we give definitions for a new device called a state-labeled finite automaton, and then we define various useful operations on these automata. (Some readers may prefer to skip Section 3 on a first reading.)

3.1 Definitions

Definition 1

A STATE-LABELED NONDETERMINISTIC FINITE AUTOMATON (SFA) is a septuple $(V, \Sigma, \lambda, \delta, S, F, e)$ where

V is a finite set, the set of STATES,

Σ is a finite set, the ALPHABET,

$\lambda \subseteq V \times \Sigma$ is the LABELING RELATION (states are labeled with subsets of the alphabet),[5]

$\delta \subseteq V \times V$ is the TRANSITION RELATION,

$S \subseteq V$ is the set of START STATES, and

$F \subseteq V$ is the set of FINAL STATES.

e is a Boolean flag that is *true* iff the null string Λ is accepted, and *false* otherwise.

5 Without loss of generality we have chosen to label states with segments (subsets of Σ), rather than with strings (subsets of Σ^*).

Before describing the execution of an SFA, we need to define COMPATIBILITY.

Definition 2
We say that a state v is COMPATIBLE with an input $\sigma \in \Sigma$ if $\langle v, \sigma \rangle \in \lambda$.

At each step in the execution of an SFA, a subset of the states is active while the remainder are inactive. An SFA begins operation with those start states active that are compatible with the first symbol in the input string. If, at a certain step in processing, a subset T of states is active, then at the next step, the subset T' of states reachable from T and compatible with the next input become active. This operation is formalized below, following the approach taken by Partee et al. (1990). First we define a SITUATION to be the processing status of an SFA.

Definition 3
A SITUATION of an SFA, A, is a triple $\langle x, T, y \rangle$ where $T \subseteq V$ is the set of currently active states, and x and y are the portions of the input string to the left and right of the reading head, respectively.

As an SFA operates, it moves through a sequence of these situations. Now $\vdash_A$ is defined as a successor relation on situations for an automaton A.

Definition 4
Let $\langle x, T, y \rangle$ and $\langle x', T', y' \rangle$ be two situations. Then $\langle x, T, y \rangle \vdash_A \langle x', T', y' \rangle$ iff there is a $\sigma \in \Sigma$ such that

(i) $y = \sigma y'$ and $x' = x\sigma$,

(ii) for each $t' \in T'$ there is a $t \in T$ such that $\langle t, t' \rangle \in \delta$, and

(iii) $\langle t', \sigma \rangle \in \lambda$ for each $t' \in T'$.

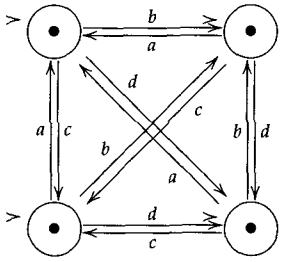
The first condition in the definition concerns σ, the tape symbol being scanned. This symbol is the first in the string y and the last in the string x'. The second condition concerns the transition relation, requiring that the new situation must be reachable from the previous situation. The third condition is a check that σ is in the label set of each currently active state. We define $\vdash_A^*$ to be the transitive closure of $\vdash_A$. Now we can specify the conditions under which an SFA accepts a string, where Λ is the empty string.

Definition 5
Let $A = (V, \Sigma, \lambda, \delta, S, F, e)$ be an SFA and let $w \in \Sigma^*$. A ACCEPTS w iff either e is true and $w = \Lambda$, or $\langle \sigma, \{s\}, \alpha \rangle \vdash_A^* \langle \sigma\alpha, F', \Lambda \rangle$, for some $\langle s, \sigma \rangle \in \lambda$, $s \in S$, $\alpha \in \Sigma^*$ and $F \cap F' \neq \emptyset$, and where $w = \sigma\alpha$.

If no string can cause an SFA to have more than one active state at any processing step, then we say that the SFA is DETERMINISTIC. In order to signify that an SFA accepts the empty string Λ (i.e., if e is true), we shall include a special state, labeled with a distinguished symbol $\emptyset$, which is marked both initial and final.

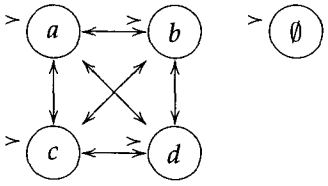
3.2 Relationship to FSAs

The equivalence of SFAs and arc-labeled finite-state automata (FSAs) follows from the equivalence of Mealy and Moore machines.[6] Although SFAs are no more expressive than FSAs, there are good linguistic reasons for wishing to use them. The primary difference between the two devices lies in the relative ease with which particular generalizations can be expressed. As an illustration, we shall consider the automaton that prohibits two adjacent occurrences of any given symbol in a string, a constraint known in autosegmental phonology as the Obligatory Contour Principle. Here is the FSA version, using the graphical conventions for representing FSAs adopted by Hopcroft and Ullman (1979). Bullet marks a state, circled states are final, and states with incoming arrow heads are initial.



The notation we use for state-labeled automata is different. Because states carry labels, there is no need to use the contentless bullet symbol to mark a state. Instead, the label itself marks the state. Initial and final states are indicated by arrowheads and circles, as for FSAs. Arcs are unlabeled.

The SFA that does not admit adjacent occurrences of any symbol in the alphabet is:



Notice that the number of labels required by the SFA is considerably smaller (5 labels) than the number required in the FSA (12 labels), while the number of transitions is the same for both. On the other hand, the SFA has an extra state, labelled with $\emptyset$ to show that the automaton accepts the null string (see the final clause of Definition 1).

The difference between the two devices becomes even clearer when we consider the representations for $a^\star$ (zero or more as) and a^+ (one or more as). First, here are $a^\star$ and a^+ expressed as FSAs.

Observe that the specification of the Kleene star requires one state, while Kleene plus requires two. If we use SFAs instead, we find the reverse: Kleene star requires two

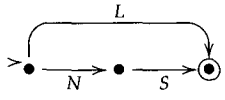
6 See Hopcroft and Ullman (1979, p. 44) for a discussion of this equivalence. An FSA is a Mealy machine that ignores its input, while an SFA is a Moore machine that ignores its input.

states, while Kleene plus only requires one.

As we shall see in Section 4, the semantics of phonological representations requires frequent use of the Kleene plus and little use of the Kleene star. The intuition behind this is simple. Recall from Section 2 that phonological entities such as distinctive features are considered to be descriptions of phonetic events that may extend over an interval of time. As we saw in the case of tone, the defining feature of an autosegment may be spread across several segments. Crucially, however, an autosegment must be present at *at least one point*, and so it makes sense to view phonological entities—such as segments and autosegments—in terms of the Kleene plus rather than the Kleene star.

It might be reasoned that our interval interpretation of segments is better pictured with arc-labeled devices. After all, the states resemble points while the arcs resemble extended intervals. Furthermore, it may be tempting to use a single arc between two temporally distant points to show the spreading of an autosegment coarticulated with two consecutive autosegments on another tier. For example, (8) shows a labial autosegment, **L**, bridging two instants also bridged by a nasal segment, **N**, and stop, **S**.

8.



However, this representation is flawed: the semantics assigned to coterminous paths contradicts the standard interpretation of FSAs. The automaton pictured above would normally be interpreted as *either* a nasal followed by a stop *or* by a single labial articulation. Crucially, it could not be interpreted as necessarily *both* a nasal followed by a stop *and* a labial articulation.

While viewing states as instants of time and arcs as intervals offers some iconicity, it is also misleading. It is just as natural—and more in keeping with the logical foundations presented in Section 2—to employ the states for temporally extended intervals, and the arcs for the relationship of immediate precedence.

3.3 Basic Operations

In this section we define the operations of concatenation, union, intersection, and complement on SFAs. These operations correspond naturally to the operations on the languages accepted by the automata, as indicated in the following table.

operation	automata	languages
concatenation	AB	$\{ab \mid a \in L(A), b \in L(B)\}$
union	$A \sqcup B$	$L(A) \cup L(B)$
intersection	$A \sqcap B$	$L(A) \cap L(B)$
complement	$\overline{A}$	$\overline{L(A)}$
Kleene plus	A^+	$L(A)^+$
Kleene star	$A^\star$	$L(A)^\star$

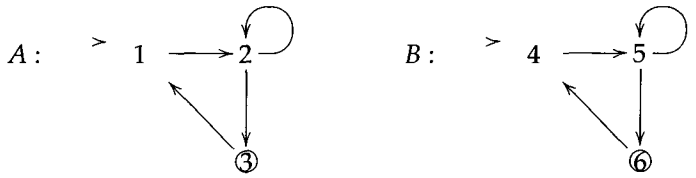
The Kleene plus operation, which, when applied to a language L gives another L^+, contains the concatenation of one or more strings from L. The Kleene star operation takes L to $\{\Lambda\} \cup L^+$ and is written $L^\star$.

Recall that the structure $(\Sigma^\star; \cup, \cap, \neg, \emptyset, \Sigma^\star)$, containing languages over an alphabet Σ together with the standard set operations, is a Boolean algebra (Partee et al. 1990, p. 297ff). Similarly, if A is the set of SFAs, then $(A; \sqcup, \sqcap, \neg, \bot, \top)$ is also a Boolean algebra, where $\bot$ is the empty automaton (i.e., $L(\bot) = \emptyset$) and $\top$ is the automaton that accepts $\Sigma^\star$ (i.e., $L(\top) = \Sigma^\star$).[7]

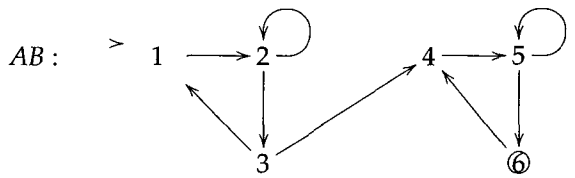
The concatenation of two SFAs A and B, written AB, has an arrow linking each final state of the first SFA to each initial state of the second. The states that are initial or final in AB depend on whether A or B accepts the empty string Λ, as specified in the following table.

A	B	AB	AB initials	AB finals
$\Lambda \notin L(A)$	$\Lambda \notin L(B)$	$\Lambda \notin L(AB)$	A initials	B finals
$\Lambda \in L(A)$	$\Lambda \notin L(B)$	$\Lambda \notin L(AB)$	A & B initials	B finals
$\Lambda \notin L(A)$	$\Lambda \in L(B)$	$\Lambda \notin L(AB)$	A initials	A & B finals
$\Lambda \in L(A)$	$\Lambda \in L(B)$	$\Lambda \in L(AB)$	A & B initials	A & B finals

Suppose we wished to recognize the language AB where $A = (12^+3)^+$ and $B = (45^+6)^+$. The SFAs describing A and B are the following.



Linking final states of A to the initial states of B gives the concatenation.

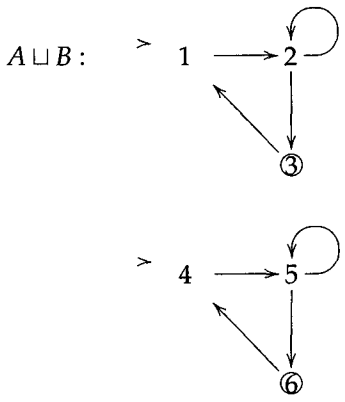


Since neither A nor B accepts Λ, the initial state of AB is the initial state of A, and the final state of AB is the final state of B.

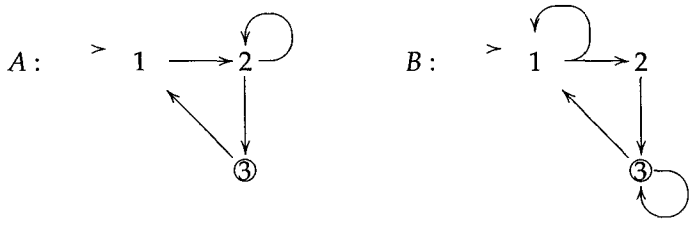
The union (or disjunction) on SFAs is similar, in many ways, to the concatenation operation. The difference is that rather than executing in sequence, the automata operate in parallel. The union of two automata A and B, written $A \sqcup B$, accepts the string s iff *either* A *or* B, or *both*, accept s. The union of A and B is expressed dia-

7 Of course, these algebras are different, for there are many languages that cannot be defined by SFAs.

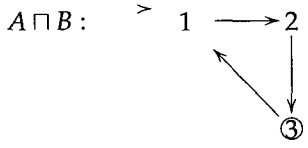
grammatically by placing the diagram for A alongside the diagram for B. The union $(12^+3)^+ \sqcup (45^+6)^+$ is drawn as:



The intersection of two SFAs A and B, written $A \sqcap B$, accepts a string s iff both A and B accept s. Consider the intersection of the automata that recognize the languages $(12^+3)^+$ and $(1^+23^+)^+$. These two automata are:



Two states are compatible if and only if their label sets have a nonempty intersection. The intersection of these automata is formed by taking all pairs of states that are compatible and linking these with arcs whenever both projections of the pairs are linked. The intersection of the above automata is shown below.



This automaton recognizes the language $(123)^+$.

The complement of an automaton A, written $\bar{A}$, accepts a string s iff A rejects s. One way of forming the complement involves the following steps. First, the automaton must be DETERMINIZED (Hopcroft and Ullman 1979, pp. 22ff). The next step is to form the COMPLETION. A complete automaton is one that has a transition from every state for each element of Σ. The final step is to mark all final states nonfinal, and all nonfinal states final. So if S is the set of states and F is the set of final states, then $S \backslash F$ is the set of final states in the complement.

3.4 Transducers

In the previous three sections, we defined state-labeled automata and some operations that can be used to combine them. We also saw that these automata are equivalent

to traditional, arc-labeled automata. Just as we can define arc-labeled automata called finite-state transducers (FSTs), we can define state-labeled transducers (SFTs).

An (epsilon free) state-labeled transducer is just an SFA with a special alphabet. Instead of labeling each state with a subset of a single alphabet, we label them with subsets of the product of two alphabets. The strings accepted are sequences of pairs consisting of one letter from each alphabet. A transducer can be used as a translator: it takes as input one half of the label on a state, and simultaneously writes as output the other half. All output strings generated by a path from initial to final states are translations of the input string recognized by the same path. Since the SFT is also an SFA, intersection is defined for SFTs.

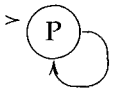
The reader may wonder whether there is any distinction between one-level phonology and two-level phonology if SFAs and SFTs are formally identical. There is an important distinction to be drawn, however. First, most two-level models employ FSTs *with epsilons*, which are more powerful devices than FSAs. Second, in the one-level model, representations and rules are interpreted as automata. In contrast, the two-level model employs strings for representations and automata for rules. Finally, in one-level phonology surface forms and generalizations about them are stated directly in a hierarchical lexicon akin to that of head-driven phrase structure grammar (HPSG) (Pollard and Sag 1987), rather than being mediated through a transducer (Bird and Klein, in press).

4. Association and Synchronization

In this section we present the automaton-based semantics for autosegmental phonology.

4.1 The Representation of Autosegments and Tiers

Recall that an autosegment denotes a possibly extended interval. In terms of automata, this means that an autosegment must allow multiple copies of its defining property. This is expressed as follows.



This state of affairs fits well with our intuitive understanding that a pair of adjacent intervals in which some property holds is indistinguishable from a *single* interval—the union of the first two intervals—during which that same property holds. Furthermore, such a claim connects with the Obligatory Contour Principle (7).[8]

Unfortunately, however, this definition of autosegment is inadequate. Suppose we have a nasal segment **N** that is lexically unspecified for its place of articulation. In a language with the nasals **m** and **n**, the intention is that this segment denotes intervals such as the following:

| m | m | m | m | m |

| n | n | n | n |

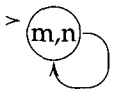
However, there is nothing to stop **N** from denoting the following interval:

| m | n | n | m | n | m |

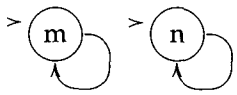
8 A consequence of this approach is that it circumvents some potential problems caused by our not employing epsilons (cf. Section 3.1). If an autosegment alternates with zero, we do not employ ϵ for the zero alternant but permit surrounding autosegments to extend to 'fill in the gap.'

Here, it is clear that **N** is behaving like a variable. If **N** is instantiated, then the entire interval must remain homogeneous. So the representation of this **N** is not (9a) but (9b).

9. (a)



(b)



Some autosegments, however, appear to lack this homogeneity property. For example, the Turkish word **peçeleri** contains several front vowels. An analysis of such words that employs the principles of vowel harmony posits a **+front** autosegment that is associated with every vowel of the word. Observe that this autosegment is heterogeneous, as it includes in its temporal extent both **e** and **i**. Accordingly, its interpretation will follow the scheme of (9a) above. Briefly continuing in this vein, we can conceive of a range of different kinds of segment, using varying numbers of states and varying numbers of labels per state. Four options are described below:

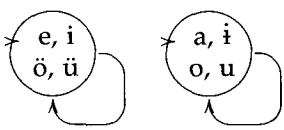
Simple Segments. These capture the ordinary kind of segment and consist of a single state labeled with a singleton set. In general, when we employ a symbol like **b** it will be interpreted as a simple segment.

Homogeneous Segments. These represent slots (like **N**) and members of templates (like **CVCCVC**), and consist of more than one state. Each state is labeled with a singleton set. An example of a homogeneous segment is found in (9b).

Heterogeneous Segments. These represent spreading autosegments, like **+high** (and **b** in Section 5.4). The automata have a single state which is labeled with a nonsingleton set. An example of a heterogeneous segment is found in (9a).

Hybrid Segments. These represent spreading autosegments that have greek letter variables, like α**place** or α**high**. An example of a hybrid segment for α**front** is given in example (10).

10.



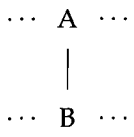
Recall that an autosegmental tier is just a linear ordering of autosegments. Therefore, if **P**, **Q** and **R** are segments (of any of the four kinds specified above), then a tier **P-Q-R** is formed by simply concatenating **P**, **Q**, and **R** together.

Now we have seen the interpretation of autosegments and tiers. The *synchronization* of tiers is controlled by association lines. The next section discusses the interpretation of these lines.

4.2 The Interpretation of Association

In Section 2 we presented an interpretation of association based on temporal overlap. Now we must find a way of simulating this temporal structure using automata. Let us begin by considering the simplest possible autosegmental diagram.

11.



Since each autosegment denotes an interval and the two intervals must overlap, we would like to interpret the above diagram as describing any of the following strings, among others:[9]

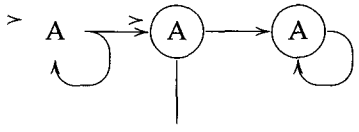
A	A	A	•	•
•	•	B	B	B

A	A	A	A	A
•	•	B	B	•

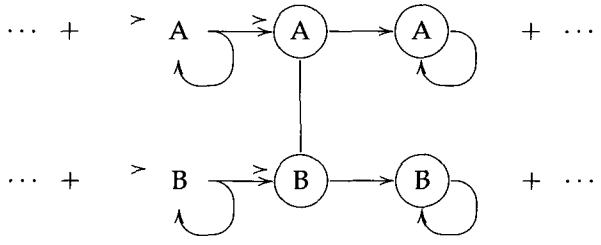
•	•	A
B	B	B

What kind of automaton will give us the required behavior? The clue is that a pair of intervals overlap if and only if they share a point in common (Bird and Klein 1990, p. 36). Note that in each of the above diagrams, the third interval contains an instance of both **A** and **B**, and the existence of this interval was both a necessary and sufficient requirement for the association line to have its overlap interpretation. So we need an automaton for each autosegment that captures the two required properties: (i) denotation of an extended period, and (ii) existence of a special point. The "automaton" required for the autosegment **A** is given in (12).

12.



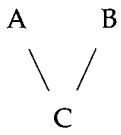
This automaton accepts any string of one or more **A**s, and requires that there is some **A** that is coincident with an autosegment somewhere else. The line extending from the middle state informally indicates that this state is simultaneous with a state in another automaton. Now we must create a similar automaton for the autosegment **B**. Recall from Section 4.1 that we need to construct tiers for **A** and **B** by concatenating the autosegments to the left and right (elided in (11)). Finally, the automata are put together as shown below:



9 Note that in using such diagrams we are attempting to isolate the effects of two automata: those cells containing both **A** and **B** should be understood as containing **A** ∩ **B**.

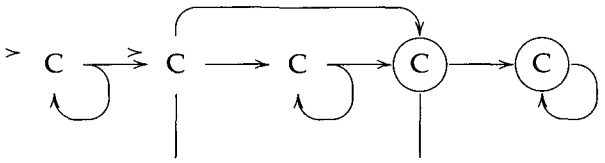
This unusual kind of automaton represents a halfway house between autosegmental diagrams and SFAs; we shall call it a SYNCHRONIZED SFA. Such automata have a simple interpretation: both component automata run in parallel, but the second state of the **A** automaton is active iff the second state of the **B** automaton is simultaneously active. In a sense, we have converted an autosegmental model involving intervals and overlap into a simpler model involving atomic periods and simultaneity. Now consider autosegmental diagram (13).

13.



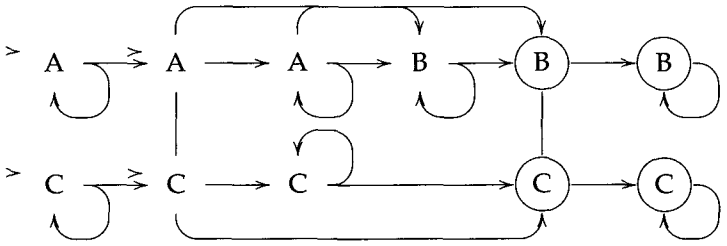
Note that there is no additional material on either side of the group of three autosegments (cf. (11)). Therefore, we assume that both tiers are descriptions of a complete utterance and so must begin and end simultaneously.[10] In this diagram, **C** has two associations. The automaton we need for **C** is more complex than before, since the interval that **C** denotes must have *two* special points, one for overlap with **A** and the other for overlap with **B**. The required automaton for **C** is given in (14).

14.



Automaton (14) is the concatenation of two copies of an automaton like (12). Putting the automata for **A**, **B**, and **C** together gives the synchronized automaton in (15).

15.



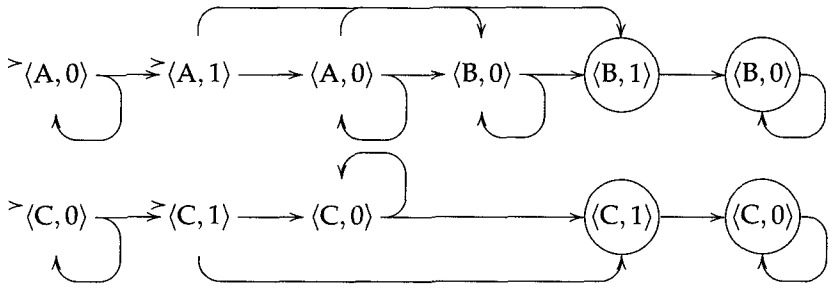
The behavior of synchronized SFAs can be simulated by an ordinary SFA, as we show in the next section.

4.3 Simulating Synchronized Automata

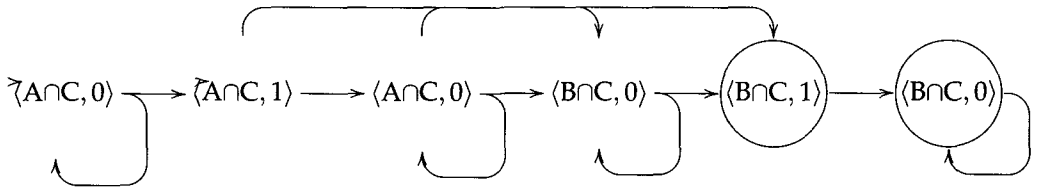
To do this simulation, we need to do away with the synchronization lines connecting the automata. Starting with (15), we add indices to each state: a 0 to unsynchronized

10 It is a trivial matter to do away with this restriction, since we can always add a completely unspecified autosegment to the start and end of each tier, thereby permitting slippage between the substantive material on each tier.

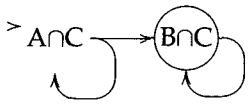
states and a 1 to synchronized states, and then erase the lines.



Observe that each of these state labels is actually a pair. These automata are defined over the product alphabet $\Sigma \times \{0, 1\}$. The intersection of these automata is as follows:



The function of the indices was to rule out certain states in the intersection. Now that the indices have served their purpose, we can erase them and further simplify the automaton:



This, then, is the semantics assigned to (13). Since we no longer require the graphical notation for synchronization, it will be convenient to represent SFAs using the notation of regular expressions over ordered pairs. In order to do this it is useful to employ some macros. An autosegment A is represented as $s(A) =_{\text{def}} \langle A, \bullet \rangle^+$. Bullet ($\bullet$) is used here as a context-dependent wildcard, indicating an alphabet. (In this definition of $s(A)$, the bullet indicates the alphabet {0,1}; in the definition of $a(n)$ below it indicates the alphabet Σ.) The n association lines incident to an autosegment are expressed as follows:

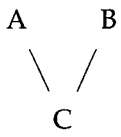
$$a(n) =_{\text{def}} \langle \bullet, 0 \rangle^\star \left(\langle \bullet, 1 \rangle \langle \bullet, 0 \rangle^\star \right)^n$$

Observe that $a(m+n) = a(m) + a(n)$. Finally, we combine these two macros into a third macro thus:

$$A{:}n \quad =_{\text{def}} \quad s(A) \sqcap a(n)$$

This states that A is an autosegment with n associations. Now we shall illustrate the workings of these definitions. Consider diagram (13) again, reproduced below:

13.



A singly associated autosegment, such as A, is written down as the following formula:

$$\begin{aligned} A{:}1 &= s(A) \sqcap a(1) \\ &= \langle A, \bullet \rangle^{+} \sqcap \langle \bullet, 0 \rangle^{\star} \langle \bullet, 1 \rangle \langle \bullet, 0 \rangle^{\star} \\ &= \langle A, 0 \rangle^{\star} \langle A, 1 \rangle \langle A, 0 \rangle^{\star} \end{aligned}$$

We can now simply write down a formula for (13):[11] $(A{:}1{+}B{:}1) \sqcap C{:}2$. This expression evaluates to the following:

$$\langle A \cap C, 0 \rangle^{\star} \langle A \cap C, 1 \rangle \langle A \cap C, 0 \rangle^{\star} \langle B \cap C, 0 \rangle^{\star} \langle B \cap C, 1 \rangle \langle B \cap C, 0 \rangle^{\star}$$

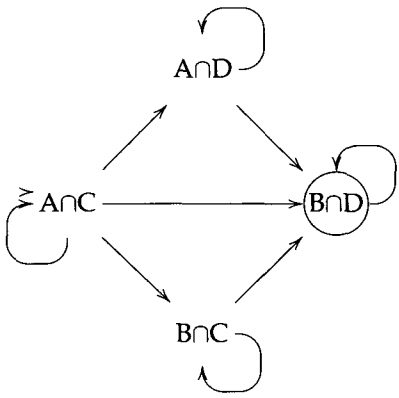
Now that the indices have served their purpose, we would like to ignore them by projecting the ordered pairs onto their first element. The result of evaluating this projection for our current example is $(A \cap C)^{+}(B \cap C)^{+}$, which is the intended interpretation of diagram (13). We shall adopt the following notational convention: if D is the encoding of a diagram then $[D]$ is the projection of the encoding that ignores the indices.

As another example, consider an autosegmental diagram consisting of two tiers, each with two autosegments, and two association lines between the tiers, as shown in (16).

16.

The expression for (16) is $(A:1 \;\; B:1) \sqcap (C:1 \;\; D:1)$. This evaluates to the following expression under the projection: $(A \cap C)^{+}((A \cap D)^{+} \sqcup (B \cap C)^{+} \sqcup \emptyset)(B \cap D)^{+}$. The corresponding automaton for (16) is given in (17).

17.



11 It is important to notice that the numerals in this expression are not the same as the indices that occur as the second member of pairs like $\langle A, 1 \rangle$. The former represent degree of association, while the latter function as synchronization marks in an automaton.

Automaton (17) will accept the following sequences, among others:

A	B
C	D

A	B	B
C	C	D

A	A	B
C	D	D

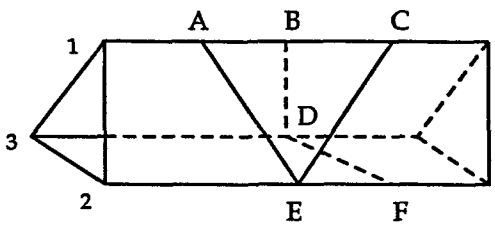
Note that all of these examples have *A* overlapping *C* and *B* overlapping *D*. The second and third examples have an extra cell, for *B* overlapping *C* and *A* overlapping *D*, respectively. This range of possibilities is compatible with (16), as it requires an *A*–*C* overlap and a *B*–*D* overlap and optionally permits an *A*–*D* overlap or a *B*–*C* overlap (but not both).

We have now seen an automaton-based interpretation of an autosegmental chart. Next we consider how the above regular expressions defined over ordered pairs can be generalized to representations consisting of more than one chart.

4.4 Multiple Charts

It is straightforward to generalize the interpretation procedure for single charts to one for representations with an arbitrary number of charts. Recall that for one chart we needed to employ ordered pairs. In general, for n charts we must employ ordered $n + 1$-tuples. The construction will be demonstrated using a diagram attributable to Pulleyblank (1986, p. 13) involving three tiers and three charts.[12]

18.



Now, since there are three charts in (18) we must employ 4-tuples. We adopt the following abbreviatory conventions:

$$
\begin{aligned}
s(a) &=_{\text{def}} \langle a, \bullet, \bullet, \bullet \rangle^{+} \\
a_{12}(n) &=_{\text{def}} \langle \bullet, 0, \bullet, \bullet \rangle^{\star} (\langle \bullet, 1, \bullet, \bullet \rangle \langle \bullet, 0, \bullet, \bullet \rangle^{\star})^{n} \\
a_{23}(n) &=_{\text{def}} \langle \bullet, \bullet, 0, \bullet \rangle^{\star} (\langle \bullet, \bullet, 1, \bullet \rangle \langle \bullet, \bullet, 0, \bullet \rangle^{\star})^{n} \\
a_{13}(n) &=_{\text{def}} \langle \bullet, \bullet, \bullet, 0 \rangle^{\star} (\langle \bullet, \bullet, \bullet, 1 \rangle \langle \bullet, \bullet, \bullet, 0 \rangle^{\star})^{n} \\
A{:}p{:}q{:}r &=_{\text{def}} s(A) \sqcap a_{12}(p) \sqcap a_{23}(q) \sqcap a_{13}(r)
\end{aligned}
$$

We can now write down the expression for (18) as follows:

19. $(A{:}1{:}0{:}0 \;\; B{:}0{:}0{:}1 \;\; C{:}1{:}0{:}0) \sqcap (E{:}2{:}0{:}0 \;\; F{:}0{:}1{:}0) \sqcap D{:}0{:}1{:}1$

The first three terms of this expression correspond to tier 1 of (18). The first term of the expression concerns the autosegment *A* and its association line on chart 1-2. The second term concerns *B* and its line on chart 1-3. Notice that lines *AE* and *BD* are in

12 Pulleyblank observes that the temporal interpretation of this diagram is ill-defined if association is assumed to be transitive. However, since overlap is not a transitive relation we do not have this problem.

different charts in (18) and so the count of association lines is in a different position in the 4-tuple for A and B. The third term concerns C and its line in chart 1-2. The fourth and fifth terms of the expression concern the E-F tier. Since E is doubly associated in chart 1-2, a 2 is used for E's degree of association. D also has two associations, but they are in different charts. Expanding and projecting this expression gives the following:

$$(A \cap D \cap E)^+(B \cap D \cap E)^+(C \cap D \cap E)^+(C \cap D \cap F)^+$$

So the shortest string that satisfies the constraints expressed in diagram (18) is the following:

A	B	C	C
D	D	D	D
E	E	E	F

Recall that the encoding of the three-tier diagram (18) was given as expression (19). In such expressions it is difficult to identify tiers. Therefore, in the remainder of this paper we shall write expressions like (19) in the following format:

20.

tier 1 $A{:}1{:}0{:}0$ $B{:}0{:}0{:}1$ $C{:}1{:}0{:}0$
tier 2 $E{:}2{:}0{:}0$ $F{:}0{:}1{:}0$
tier 3 $D{:}0{:}1{:}1$

In general, if D is a diagram, then $\mathcal{E}(D)$ is its encoding in the format exemplified in (20).

4.5 Evaluating the Encoding

Now, we evaluate our encoding with respect to Kornai's desiderata: computability, compositionality, invertibility, and iconicity (Kornai 1991).

Computability: The number of terms in the encoding is equal to the number of autosegments, and each term has a fixed size.[13] Therefore, the encoding can be computed in linear time.

Compositionality: If D_1 and D_2 are two autosegmental diagrams then $\mathcal{E}(D_1D_2) = \mathcal{E}(D_1)\mathcal{E}(D_2)$, where concatenation of encodings is done in a tier-wise manner. Thus the encoding is compositional.

Invertibility: A representation can be reconstructed from its encoding.

Iconicity: If an autosegment in a diagram is changed, the effect on the encoding is local, since only one term is altered. However, if an association line is added or removed, two terms must be altered. Although these terms may not be adjacent, we believe the encoding is more iconic than Kornai's triple code (see Section 5.2), in which changes can affect an unbounded amount of material.

In addition to these properties, our encoding can be used *directly* as a finite-state recognizer of surface forms, simply by forming the intersection of the n tier encodings

13 Observe that when we expand these macros the resulting expression has $s + 2a$ terms, where s is the number of autosegments and a is the number of associations.

and projecting the first elements of the tuples. Note that if we revert to the encoding in (19), where the tier encodings are combined into a linear expression using intersection, then compositionality is lost. Thus, the encoding is either linear or compositional, but not both. Unfortunately, this is the best that we can hope for; Wiebe (1992) has shown that a compositional linear encoding does not exist.

4.6 Phonological Rules

Phonologists typically encode their descriptive generalizations in terms of RULES. Often these rules are interpreted as processes that manipulate representations. In the one-level approach they are interpreted as a logical implication between two descriptions, which simplifies to a single description given the Boolean operations presented in Section 3.3.

As our first example, consider the phenomenon of homorganic nasal assimilation, whereby nasals agree in place of articulation with the following consonant. An SPE-style rule for this is given in (21a), the corresponding logical implication in (21b).

21. (a) [+nasal] $\rightarrow$ [αplace] / __ [+cons,αplace]
    (b) [+nasal][+cons] $\rightarrow$ [αplace][αplace]

Thus, the sequences **mb** and **nd** are allowed, while **md** and **nb** are ruled out. Let **N** = {**m**,**n**}, **S** = {**b**,**d**}, **L** = {**m**,**b**}, and **A** = {**n**,**d**}. The required constraint can be expressed as **NS** $\rightarrow$ **LL** $\sqcup$ **AA**. However, in order to make this rule apply to a whole string (rather than just the first **NS** sequence it comes across), we must express it in the following format.[14]

22. $\neg(\bullet^\star(\mathrm{NS} \sqcap \overline{\mathrm{LL} \sqcup \mathrm{AA}})\bullet^\star)$

This states that it is not possible to find anywhere a nasal-stop cluster (**NS**) that is not made up of two labials (**LL**) or two alveolars (**AA**). We can simplify the above expression to $\overline{\bullet^\star(\mathrm{mA})\bullet^\star} \sqcap \overline{\bullet^\star(\mathrm{nL})\bullet^\star}$.

Now consider a general rule of the form SD $\rightarrow$ SC. Since SD and SC pertain to *parts* of a string rather than a whole string, we have to ensure that the rule applies to all substrings of an 'input' string *S*. We do this as follows:

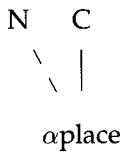
23.
$$\begin{aligned} & \forall s \subseteq S, \mathrm{SD}(s) \rightarrow \mathrm{SC}(s) \\ \Rightarrow\ & \forall s \subseteq S, \neg\mathrm{SD}(s) \vee \mathrm{SC}(s) \\ \Rightarrow\ & \neg\exists s \subseteq S, \mathrm{SD}(s) \wedge \neg\mathrm{SC}(s) \\ \Rightarrow\ & \neg(\bullet^\star(\mathrm{SD} \sqcap \overline{\mathrm{SC}})\bullet^\star) \end{aligned}$$

This is how we arrived at (22) from (21b).

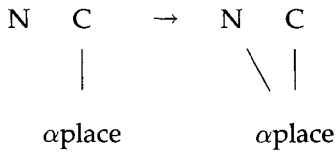
Autosegmental rules can also be expressed in this framework. Consider again the case of assimilation. The following diagram is the autosegmental rule corresponding to the SPE rule in (21). Here, α**place** is a hybrid autosegment (see Section 4.1) ranging over places of articulation.

14 Note that we employ a '$\neg$' sign or an overline to represent complement, depending upon which is most convenient.

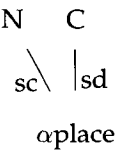
24.



This rule states that wherever an **NC** sequence can be found, if the **C** is associated with an **L**, then the **N** is also associated with **L**. We can express this rule in the more familiar rewrite notation:



We can give an automaton-based semantics to this rule. In order to do this, we must employ *two* independent charts between the two tiers, one for the structural description and one for the structural change. We can represent this as follows, where 'sd' refers to lines in the structural description chart, and 'sc' refers to lines in the structural change chart.



Now we can write down the formulas for the structural description and the structural change independently and combine them into the rule format of (23).

25. $$\neg\left(\bullet^{\star}\begin{bmatrix} \bullet{:}0^{\star} & \text{N:0} & \text{C:1} & \bullet{:}0^{\star} \\ \bullet{:}0^{\star} & \alpha\text{place:1} & & \bullet{:}0^{\star} \end{bmatrix} \sqcap \neg \begin{bmatrix} \bullet{:}0{:}0^{\star} & \text{N:0:1} & \text{C:1:0} & \bullet{:}0{:}0^{\star} \\ \bullet{:}0{:}0^{\star} & \alpha\text{place:1:1} & & \bullet{:}0{:}0^{\star} \end{bmatrix}_{sc}\Big]_{sd} \bullet^{\star}\right)$$

In (25) the 'sd' and 'sc' subscripts on the brackets refer to projection functions that ignore the structural description and structural change charts, respectively. So, in evaluating (25) we intersect the two tiers of the structural change part of the rule, and then delete the second index of each tuple. The complement of this automaton is then intersected with the structural description part of the rule and the first index of each tuple is then deleted. The final step is to add the $\bullet^{\star}$ wildcards and form the complement again. The result is an automaton that rejects any nonhomorganic **NC** clusters.

A more complex example of a phonological rule, this time concerning vowel harmony, will now be discussed. Since the advent of autosegmental phonology, vowel harmony has been analyzed as the spreading of autosegments from left to right. Here we show how such a rule can be translated into a regular constraint on surface forms.

Turkish exhibits two orthogonal types of vowel harmony: one requiring that consecutive vowels agree in fronting, the other requiring that consecutive vowels agree in rounding unless the second vowel is low. As the rounding harmony is the more complex of the two, we will take it as our example. To avoid the complications of fronting harmony, we will only consider examples involving back vowels.

Turkish has eight vowels. Four of them (**a e ı i**) are unrounded, and four (**o ö u ü**) are rounded. Turkish is an agglutinating language, and the vowels in many affixes depend on the final vowel in the root of the word. Compare, for instance, three of the cases of the words **son** *end* and **adam** *man* displayed in (26).

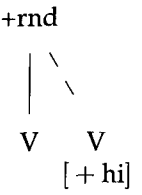
26.

case	son	adam
nominative	son	adam
accusative[15]	sonu	adamı
dative	sona	adama

When the rounded root vowel is followed by a high vowel in a suffix, this vowel must agree with the root in rounding. Low vowels in harmonising suffixes are, however, always unrounded.

The notation of autosegmental phonology makes it easy to state this generalization as a rule. The rule spreads a rounding autosegment onto the next vowel if (and only if) the next vowel is high.

27.



This rule only applies on the vowel tier, skipping over consonants, and the interpretation given to autosegments on this tier must reflect this. We use the idea of heterogeneous autosegments (see Section 4.1) to differentiate ordinary segments from autosegments on restricted tiers such as the vowel tier. A vowel on this tier may denote not only a vowel but also consonants interspersed within this vowel as well. Whereas a segmental **a** corresponds to an automaton with a single state having a singleton label (this is what we have called a SIMPLE segment; see Section 4.1), an **a** on the vowel tier corresponds to a single state automaton whose state accepts not only **a** but any consonant as well (a HETEROGENEOUS segment).

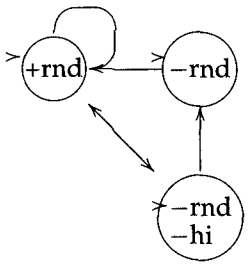
Given that the vowel tier uses this kind of heterogeneous representation, we can combine the charts of our rule into the regular expression in (28).

28.
$$\neg\left(\bullet^{\star}\left[\begin{array}{cccc} \bullet{:}0^{\star} & +\mathrm{rnd}{:}1 & \bullet{:}0^{\star} & \\ \bullet{:}0^{\star} & \mathrm{V}{:}0 & +\mathrm{hi}{:}1 & \bullet{:}0^{\star} \end{array}\right] \sqcap \neg\left[\begin{array}{cccc} \bullet{:}0{:}0^{\star} & +\mathrm{rnd}{:}1{:}1 & \bullet{:}0{:}0^{\star} & \\ \bullet{:}0{:}0^{\star} & \mathrm{V}{:}0{:}1 & +\mathrm{hi}{:}1{:}0 & \bullet{:}0{:}0^{\star} \end{array}\right]_{\mathrm{sc}}\Big]_{\mathrm{sd}} \bullet^{\star}\right)$$

Using the procedure of Section 4.6, we can translate this into a regular expression that

15 More precisely, the accusative case is used only for definite objects.

implements rounding harmony as a constraint on surface forms. After simplification, this automaton is:



Each of the states accepts the specified class of vowels and any consonant. Informally, this automaton will accept any sequence of vowels except those in which a round vowel is followed by an unrounded high vowel. This is precisely the intended effect of the autosegmental version of the harmony rule (27). This concludes our discussion of vowel harmony.

A comment is in order here about why two charts were required for the encoding of autosegmental rules. After all, our use of an input and an output chart appears to be a procedural device, and we have eschewed these from the outset. However, note that it is not possible for our structural description and structural change to be encoded on the same chart: the structural change has an association line not present in the structural description, and so they are incompatible. Of course, the *overlaps* described by the structural description and structural change are mutually compatible; it is only the *associations* that are not. Once the structural change has been computed, we can throw away the 'sc' chart, and once the structural description and structural change have been combined, we can also throw away the 'sd' chart. In this way, the rule functions only as a filter on surface forms, and there is no way for two separate rules to communicate via these rule-internal charts.

This approach permits us to interpret any nondestructive autosegmental rule.[16] Those rules involving deletion of autosegments or association lines must be approached in a completely different way. Rather than deleting an element in a particular context, we set up an alternation with zero, following Bloomfield (1926). See Bird and Klein (in press) and Bird (in press) for detailed examples of this approach to deletion.

This concludes our discussion of the automaton-based semantics for autosegmental representations and rules. In the next section we review some other attempts to treat autosegmental phonology using finite-state techniques.

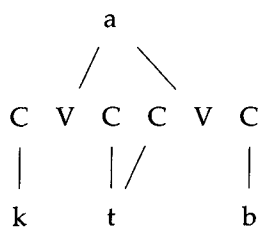
5. Other Finite-State Approaches to Nonlinear Phonology

5.1 Kay (1987)

The earliest treatment of autosegmental phonology in a finite-state setting is attributable to Kay (1987). It is tailored to the framework of nonconcatenative morphology developed by McCarthy (1981) in which consonants and vowels are segregated onto different tiers, as shown for the Arabic verb stem **kattab** in (29).

16 Applied to a destructive autosegmental rule, the interpretation determines the restriction imposed on surface forms by the rule, were it the last rule in a derivation.

29.



This verb stem contains three morphemes. Together, they mean 'caused to write.' The challenge posed by Arabic morphology is to come up with a simple account of the interleaving of the morphemes, relating the forms **a**, **CVCCVC**, and **ktb** to the stem **kattab**.

Kay's solution, which we sketch here, involves the use of a kind of transducer that reads four-tuples (rather than pairs like a normal FST). This transducer scans four strings, one for each of the three tiers in (29) and one for the corresponding surface form. In this way, Kay has identified tiers with tapes. The association relation is encoded in the way the transducer scans these four tapes.

Kay specifies a transducer by providing a set of FRAMES. In effect, each frame specifies an association between a CV-tier slot and a melodic unit, such as a **k** or an **a**. This melodic unit appears on the current surface tape cell. Each frame is a four-tuple whose components correspond to (i) the consonantal root, (ii) the CV-tier, (iii) the vocalic melody, and (iv) the surface tape. A simple frame is the following:

$$k : C : \epsilon : k$$

This frame specifies that when a **k** is being read from the consonantal tape and a **C** is being read from the CV tape, then there is an empty transition on the vocalism tape and the surface tape must have a **k**. There is a similar frame for each consonant. A frame for the vowel **a** is ϵ:**V**:**a**:**a**.

More work is necessary in order to capture the idea of autosegmental spreading. Kay modifies the transducer model so that tape symbols can be inspected without the reading head being advanced. Three notational devices manipulate the way this revised model behaves. Square brackets around one of the components of a frame causes the corresponding tape cell to be scanned without advancing the read head. Braces around a frame component behave in the same way, but only if they are scanning the final symbol on a tape. (This is required to prevent the spreading of **i**, which will not be discussed here.) Finally, the symbol **G** is used instead of **C** for geminate consonants. We shall see how these devices operate using a worked example, in which the surface form **kattab** is derived from the three lexical forms of diagram (29).

Display (30) gives the initial configuration. The box shows the collection of symbols currently being scanned on the four tapes. To the far right is the appropriate frame. An empty pair of brackets is equivalent to ϵ.

30.

k	t	b				k
C	V	G	C	V	C	C
a						[]
k						k

The first CV tape symbol is a **C** and so the current consonant **k** is written onto the surface tape and the read heads are advanced. Notice that the read head for the vowel tape is left on the first cell. This is because the frame specified that there be no transition on this tape.

31.

k	t	b				[]
C	V	G	C	V	C	V
	a					{ a }
k	a					a

In (31), the CV tape symbol is a **V**, and so the **a** is copied to the surface tape. Since this **a** is given in braces in the frame, there is no movement of the read head.

32.

	k	t	b			[t]
C	V	G	C	V	C	G
		a				[]
k	a	t				t

Having reached configuration (32), we read a **G**, which indicates a geminate. The **t** is written to the surface tape and the read head for the consonant tape stays where it is. The brackets around **t** in the frame mean that there is a nondeterministic choice between moving the read head and leaving it in the same position. However, the **G** symbol requires that the head stays put. Next we get a **C** on the CV tape.

33.

		k	t	b		[t]
C	V	G	C	V	C	C
			a			[]
k	a	t	t			t

After (33), the consonant tape read head advances[17] and the CV tape head moves on to deal with another vowel (34).

17 Note that there is nondeterminism hidden here again. As it happens, if the consonant tape does not advance then the **b** will never be used (i.e., we will get ***kattat**) and the transducer will fail, because of the requirement that all read heads be at the end of their input for a successful completion.

34.

			k	t	b		[]
	C	V	G	C	V	C	V
					a		{a}
	k	a	t	t	a		a

Finally, the consonant **b** is transferred to the surface tape.

35.

				k	t	b	[b]
	C	V	G	C	V	C	C
						a	[]
	k	a	t	t	a	b	b

The result on the surface tape is **kattab** as required. Of course, the transducer also works for recognition. We could specify a surface tape and leave one or more of the lexical tapes unspecified.

Kay's system is ingenious and we have not been able to demonstrate all of its capabilities in this small space. Nevertheless, we believe it suffers from a number of problems. The first concerns the form **katab** (form I) and the (corresponding) reflexive form **ktatab** (form VIII) with an infixed **t**. Indeed, more than half of the forms that Kay cites have infixes, although these are not explicitly analyzed in the model. Two conservative extensions to the model that encompass this infixation are (i) to insert infixes into the CV tape directly (so form VIII is **CtVCVC**) or (ii) to introduce another CV tape symbol **A** (for affix), which directs the transducer to read from a fifth tape.

A second set of problems concerns the appropriateness of Kay's model for nonlinear phonology more generally. First, notational devices like **G** move the frame 'language' away from what it is supposed to represent, namely autosegmental structures. No longer is gemination represented by association (perhaps derived by a spreading rule), but by a special marking on the skeleton. Second, the model builds in the assumption that each morpheme appears on a separate autosegmental tier. However, most applications of autosegmental phonology employ morphemes with phonological information arrayed on more than one tier (e.g., Clements and Ford 1979). Similarly, the modeling of subsegmental feature geometry of the kind advocated by Clements (1985) and others also involves a single morpheme having material on several tiers. Third, the model breaks down in the area of MORPHEMIC SEGREGATION. Since there is no principled upper bound on the number of morphemes that may be overlaid in the way McCarthy advocates for Arabic, there is similarly no principled upper bound on the number of tapes Kay's transducer would require. The assumption that each morpheme defines its own set of tiers, implicit in early work (McCarthy 1981) but explicit in more recent work (McCarthy 1989), is incompatible with a fixed upper bound on the number of tapes. Finally, using Kay's model for recognition would lead to much nondeterminism in positing **G** symbols, brackets, and braces. For example, in processing **kattab** the lexical tapes **kttb**, **CVCCVC**, and **aa** could be generated. The model generates all possible violations of the Obligatory Contour Principle.

5.2 Kornai (1991)

Kornai (1991) has developed a linear encoding of autosegmental representations that allows the two-level transducer model to be applied to autosegmental phonology. We shall present Kornai's central innovation and describe a few of its strengths and weaknesses.

As we saw in Section 4.5, Kornai presents four criteria under which an autosegmental encoding should be judged. An encoding should be easily *computable*; ideally by finite automata. It should also be *invertible*. An encoding should be *iconic*; minimally changing the input should minimally change the output. Finally, it should be compositional, in the sense that the concatenation of the encodings of A_1 and A_2 ought to be the same as the encoding of the concatenation of A_1 and A_2. Kornai demonstrates that an optimal encoding under these criteria does not exist, and he sets about defining an encoding that is claimed to be near-optimal.

We shall only cover one of the codes he considers, namely, the TRIPLE CODE. This code represents two tiers of autosegments and a chart between them as a linear description. There are four keywords in the code that are interpreted as instructions to a device that is scanning two tiers (left to right) and drawing association lines between certain pairs of segments. These keywords are as follows:

0 : leave the current segments unassociated and advance the read head on each tier;

1 : draw an association line between the current segments on each tier and advance the read heads on each tier;

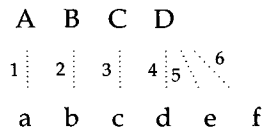
t : override the advance instruction on the bottom tier, i.e., only advance the read head on the top tier; and

b : only advance the read head on the bottom tier; retain the same segment on the top tier.

Each 0 or 1 is flanked by a statement of the current segments on the two tiers. A number flanked by two segments forms the TRIPLE that gives the code its name.

Where this code could give a number of different representations of the same autosegmental structure, Kornai (1991) restricts the encoding to operating in the same manner as the association convention of Goldsmith (1976). Association, or the lack of association, is marked left-to-right in a one-to-one fashion until one tier is devoid of new autosegments. From that point, only one tier advances until all remaining autosegments are represented in the linearization.

As examples, let us encode two charts, the first completely devoid of associations. To show the pairs of current autosegments through the steps of the encoding, we link them with dotted lines indexed by the count of the step in the derivation.



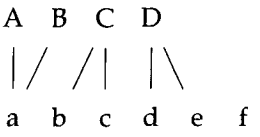
The first four code steps give the following encoding (separating triples with full stops):

```
A0a.B0b.C0c.D0d
```

For the remaining two steps, we must spread (in virtual associations, not real ones) the final segment of the top tier, remembering to record the fact that only the read head on the bottom tier advances. The total encoding of this chart is thus the following:

```
A0a.B0b.C0c.D0d.b.D0e.b.D0f
```

As a second example, let us fill the same chart with some associations.



The encoding for this chart is the following:

```
A1a.t.B1a.C1b.b.C1c.D1d.b.D1e.b.D0f
```

This code can be measured against the four criteria given above. The code is definitely computable; we have just given an algorithm for constructing the code for an arbitrary chart. Likewise the code is invertible: given any code, it is clearly possible to reproduce the original chart. Similarly, given the chart decoded from any encoding, it is likewise possible to reproduce the encoding.

The triple code is, however, neither iconic nor compositional. Consider the two autosegmental representations given below.

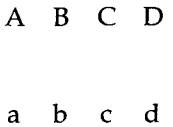
The triple code for the first representation is `A0a.B0b.C0c`. For the second representation, the triple code is the sequence:

```
A0a.b.A0b.b.A1c.t.B0c.t.C0c
```

As we can see, a minor change in one part of the autosegmental representation has resulted in major changes in much of the triple-coded representation. Thus the code is not iconic.

Similarly, if we concatenate the two representations shown below

we get



If we concatenate the two encodings, we get *A0a.b.A0b.b.A0c.B0d.t.C0d.t.D0d*, which is not the same as the encoding of the concatenation: *A1a.B1b.C1c.D1d*. So the triple code satisfies only two of the desiderata Kornai gives for linearizations of autosegmental representations. As mentioned above, Wiebe (1992) has proven that linear encodings cannot get much better than this: no linear encoding can be both invertible and compositional.

Under the triple code, a lot of phonological processes are not finite-state. For example, consider the process of putting morphemic tiers together. In the case of the Arabic example we saw in (29), we need to combine the root morpheme (with empty template and vocalic tier) **ktb** and the template (with empty vocalic and consonantal tiers) **CVCCVC** together in such a way that we end up with the encoding in (38). In autosegmental representations, this is achieved by concatenation, and the "before" and "after" diagrams appear in (36) and (37).

36.

C V C C V C

k t b

37.

C V C C V C

k t b

Concatenation of codes does not bring the same result. Display (38) shows the two concatenands, written with ∅ as place holder for the empty tier.[18]

38. C0∅.V0∅.C0∅.C0∅.V0∅.C0∅ + ∅0k.∅0t.∅0b
≠ C0k.V0t.C0b.t.C0b.t.V0b.t.C0b

To obtain the correct result, we need a more complex operation than concatenation—one that cannot be performed by any finite-state device.

If the **ktb** is read before the **CVCCVC** is read, it is necessary to have sufficient state information to be able to store **k**, **t**, and **b**, which in an alphabet of about 28 consonants involves at least $log_2(28^3) \approx 15$ extra bits of information in each state, compared to an otherwise equivalent transducer. Suppose now that we wish to apply the well-formedness condition (5), the association convention (6), and some other rules to (38) to achieve the pattern of association between the two tiers given in (29), repeated below. We are not interested here in the detail of how a transducer might perform this operation, only whether a transducer *can* perform it.

39.

C V C C V C

k t b

18 As far as we have been able to find, Kornai (1991) does not discuss how to code a representation that contains an empty tier. Here, we have filled empty tiers with a 'place holder' segment, and then used the triple code.

The encoding of (39) is C1k.V0t.t.C1t.t.C1t.V0b.t.C1b. Observe that the **t** is met in the second triple of the right-hand side of (38) and it must be 'stored' until the fourth triple in the output. In a form like **tadaḥraj**, two consonants must be stored in this way, leading again to an explosion of state information.[19] If the template consisted of n CV syllables, and the consonant tier consisted of n consonants, then the association mechanism would need to store at least $\lfloor (n-1)/2 \rfloor$ autosegments. This is, in principle, unbounded, and so the association mechanism for coded autosegmental representations is not finite state.

In general there is no limit on the amount of information that needs to be preserved as state information in the transducer. Consequently, using the triple code, certain phonological processes cannot be modeled using a finite state transducer. Indeed, this is true of any linear code (Wiebe 1992). It is true that adding each individual association—that is, finding the next position to associate, making the association and shuffling the lower tier along—can be performed by a finite-state transducer. But no principled bound on the length of the derivation can be made, and the quantity of memory required is an increasing function of the length of the derivation. Consequently, the step from unassociated to associated form cannot be made by a single FST.

This type of problem with manipulating coded representations is not limited to Arabic morphology. In fact, the problem arises whenever there is an autosegment that has a restricted set of 'landing sites' (such as the consonants or the high vowels), a very common occurrence. Another example is that of accent systems: Kornai and Kálman (1988, p. 185), citing Goldsmith (1982), state the Basic Tone Melody Association rule for Ci-Ruri, a Bantu language of Tanzania, as follows:

40. **Basic Tone Melody Association Rule:**
Associate the accented element of the Basic Tone Melody to the accented element of a word.

In general, the accented element of a word may be an unbounded distance from the start and end of a word. Goldsmith (1982, p. 53) gives the following example:

41.

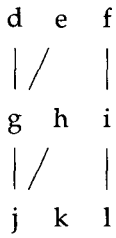
na	a	kam	ir*	e	*I milked*
			|		
L			H*	L	

As before, performing the association using the triple code requires a multiplication in state information, which in the general case cannot be bounded. True, this example only requires the automaton to store H; but Goldsmith (1982, p. 55) gives examples where there are two stresses in the word, and, therefore, the association automaton needs to store more tones (see Wiebe 1992, pp. 109–110 for the details).

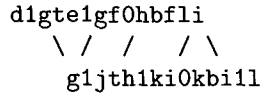
Next, consider the problem of generalizing from one chart to an arbitrary number

19 Wiebe (1992, p. 112) points out that the final consonant does not need to be stored, as it is already associated with the final C position.

of charts. Kornai (1991, p. 70) gives the following example:



Each chart is encoded as a string and then these two strings are linked by association lines. Each line indicates that the segments it connects are actually the same segment (in the original diagram). Here is the first step of the encoding of the above diagram:[20]

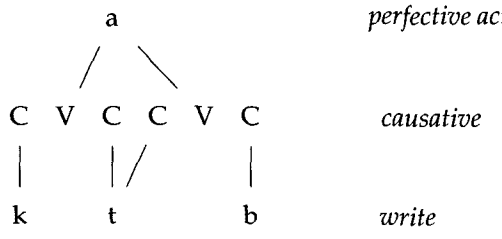


Next, this encoding can itself be encoded into a string 77 characters long. Kornai himself admits (p. 72) that his encoding is impractical for autosegmental structures having more than two charts.[21] Kornai discusses another encoding, which involves transducers that can read three (or more) tapes simultaneously. However, this comes in for the same criticism leveled at Kay's system above, namely that there is no principled upper bound on the number of tiers due to such considerations as complex feature geometry and morphemic segregation.

5.3 Wiebe (1992)

Wiebe (1992) has recently devised an encoding of autosegmental diagrams that overcomes many of the problems of Kornai's triple code. It is called the MULTI-LINEAR CODE. Consider again diagram (29), reproduced below.

29.



This diagram contains two charts. The upper chart, which connects the vowel to two **V** slots will be referred to as chart 1, and the other chart as chart 2. Since there is a total ordering on the associations in a given chart (Goldsmith 1976, p. 29), it is sufficient to record how many associations each autosegment has on each chart, without specifying where these associations lead. The multi-linear code for (29) is displayed in (42).

20 Here the dots separating the triples have been omitted to enable more convenient location of association lines.

21 The encoding procedure would not terminate for Pulleyblank's example (18). This is because the first application of the encoding would take the three-tiered structure and produce another (more complicated) three-tiered structure. In fact, this is just a special case of a more general problem, for the encoding fails for any tier structure containing a cycle, such as the one proposed by Clements (1991).

42. a11
C2V1C2C2V1C2
k2t22b2

Here, a numeral n following an autosegment **A** indicates that **A** has an association on chart n. These numerals can be stacked up; the first line specifies that **a** has *two* associations on chart 1. A given tier can participate in two charts; the second line of (42) has **C**s associated on chart 2 and **V**s associated on chart 1. Wiebe shows how the multi-linear code satisfies the criteria for computability, invertibility, iconicity, and compositionality.

Wiebe's encoding has some similarities to our revised encoding presented in Section 4.5. Here is our encoding of (29):

43.

tier 1			a:2:0:0			
tier 2	C:0:1:0	V:1:0:0	C:0:1:0	C:0:1:0	V:1:0:0	C:0:1:0
tier 3			k:0:1:0	t:0:2:0	b:0:1:0	

Suppose that $a{:}p{:}q{:}r$ is an arbitrary 4-tuple of the kind in (43). We use position in the tuple to specify which chart an association line is in, and use numerals to specify the number of association lines. However, Wiebe uses numerals to specify the chart and repetitions to specify the number of lines. There is a mapping between terms like $a11$ in Wiebe's encoding and $a{:}2{:}0{:}0$ in our encoding:

$$f(a{:}p{:}q{:}r) = a1^p2^q3^r$$

If we apply f to each term in (43), the result is as follows.

$$\begin{array}{c} a11 \\ C2V1C2C2V1C2 \\ k2t22b2 \end{array}$$

Now it should be clear that there is an isomorphism between the two encodings.

Wiebe goes on to show how his encodings can be processed by new devices called MULTI-TAPE SFAs and SFTs,[22] where each tape corresponds to a row in the multi-linear code. The devices are used for checking well-formedness constraints and applying (possibly destructive) autosegmental rules. Wiebe also demonstrates that these devices are more powerful than FSTs without epsilon transitions, claiming that they can recognize some (strictly) context-sensitive languages. He argues that this extra computational power is crucially required for processing autosegmental analyses with feature- or structure-modifying rules.

> The read heads can scan n-tuples separated by arbitrary distances, and each head reads one co-ordinate of the n-tuple under it. ... It is precisely this ability to scan different parts of an input word at the same time that is so important in modelling autosegmental rules. Association lines can associate segments in any part of one tier to segments in any part of the facing tier. In order for any computational

22 Wiebe borrows the terms SFA and SFT from an earlier version of this paper (Bird and Ellison 1992).

> device to efficiently process autosegmental representations, it must be able to scan two *associated* segments from *widely separated* parts of the representation at the same time (Wiebe 1992, pp. 95–96, *emphasis added*).

While this is a reasonable statement regarding any model like Kornai's, it does not apply to our one-level model since the notion 'widely separated' is meaningless in this context. Two terms in a multi-linear code are widely separated if they differ significantly as to their distance from the left- or right-hand end of the encoding. Thus, the longer and more slanted the line that associates the autosegments, the greater their separation. Although this is true of the ink on a page, our semantics pays no attention to the angle and length of association lines. If a pair of autosegments is associated then they are ipso facto proximate. The temporal extent of autosegments expands or shrinks to accommodate these temporal constraints imposed by association lines. The intensional character of our approach gives rise to a flexibility of interpretation that obviates the need for more powerful devices of the kind advocated by Wiebe.

Now that we have reviewed some details of other finite-state approaches to autosegmental phonology, we present a one-level analysis of an aspect of Arabic verb morphology.

5.4 A One-Level Analysis

As an alternative to either the elaboration of the transducer or the reduction of the nonlinear representation to a code, we present a partial analysis of the Arabic verb in terms of SFAs.[23] Rather than trying to generate one representation from another, we construct a description of the individual Arabic word by taking the intersection of the SFAs describing each of the morphemes. The forms we analyze here, like those given in McCarthy (1981), do not include inflections and also assume that the following suffix is consonant-initial.[24] We shall derive the stem **kattab** from three morphemes, specifying the form, the root, and the voice/mood.

The form I SFA (44) generalizes over all verbs: it accepts/generates all correct verbs of this form, as well as a number of nonsense verbs, and factors out information specific to the individual words.

44. C:1 V:0 C:1 V:0 C:1

The indices mark associations that will link this tier to the root.

Another tier is the consonantal tier. It contains heterogeneous autosegments corresponding to consonants. These symbols are heterogeneous because they allow not only the corresponding consonant, but also any of the vowels. So the **f** autosegment on the consonant-tier denotes any sequence of the segments **f, a, i,** or **u**. The two uses of **f** are disambiguated by reference to the tiers on which they occur. Under this definition, autosegments on the consonant tier can spread over vowels to the next consonantal position.

The root SFA generalizes over all stems constructed from the same root. The encoding on the consonant tier describing the root **fʕl**, *to do*, appears in (45).

23 The reader interested in other finite-state models of Arabic phonology is directed to the work of Narayanan and Hashem (1993), Beesley, Buckwater, and Newton (1989), and Beesley (1990). These have not been discussed at length here, because they do not seek to implement autosegmental phonology.

24 Biliteral roots in forms I, III, IV, V, VI, VII, VIII, X, XI, XII, and XIV, triliteral roots in forms IX and XI, and quadriliteral roots in form QIV reorder the consonant-vowel sequence in the final syllable of the root if followed by a vowel-initial inflection. Where possible, the Arabic verb stem will metathesize or delete short vowels to create a geminate with root consonants (McCarthy 1981, pp. 197f).

45. •:0* f:1 (•:0* ʕ:1)+ l:1

The wildcards are necessary for affix consonants, such as the **n-** prefix of form VII or the **-t-** infix of form VIII.

It might be argued that fixing the associations (by the indices) in the specification of the morpheme is redundant—that the associations should be supplied by rule. However, 3 of the 15 triliteral forms of the Arabic verb, namely forms II, V and XII, violate the association convention, multiply associating central autosegments rather than peripheral ones. For these forms there seems little choice but to lexically specify the associations or else introduce an ad hoc, morphologically triggered, rule (McCarthy 1981, p. 392, rule 24). In our notation, we can specify these forms as:

46.
II	C:1 V:0 C:1 C:1 V:0 C:1
V	t:1 V:0 C:1 V:0 C:1 C:1 V:0 C:1
XII	C:1 C:1 V:0 w:0 C:1 V:0 C:1

The intersections of the three forms in (46) with the root automaton (45) are:

47.
II	f:1 V:0 ʕ:1 ʕ:1 V:0 l:1
V	t:1 V:0 f:1 V:0 ʕ:1 ʕ:1 V:0 l:1
XII	f:1 ʕ:1 V:0 w:0 ʕ:1 V:0 l:1

The vocalism for the active perfect verb (except form I) is very simple to express: it is a sequence of **a**s interleaved with consonants. On a vowel plane, where each autosegment is heterogeneous, accepting a vowel and any of the consonants, this aspect may be expressed by the single autosegment **a**, which corresponds to the automaton in (48).

48.

Forming the intersection of this SFA with the intersections of roots and forms shown in (47) gives the following forms.

49.
II	f:1 a:0 ʕ:1 ʕ:1 a:0 l:1
V	t:1 a:0 f:1 a:0 ʕ:1 ʕ:1 a:0 l:1
XII	f:1 ʕ:1 a:0 w:0 ʕ:1 a:0 l:1

This brief analysis shows the ease with which a computable analysis of nonconcatenative morphology can be constructed in the SFA formalism. While it may not cover all possible generalizations about Arabic verbal structure, the important point here is that it captures generalizations that are always surface-true of the phonology of the separate morphemes that come together to build up the verb.

5.5 Automata versus Linearizations

We have seen that Kornai (1991) finds it necessary to choose between the imposition of restrictions on autosegmental phonology and the loss of finite stateness in the transduction relationship. As it turns out, the one-level approach does not suffer from this problem. In this section we explain why.

Note that the natural processes by which finite-state automata are combined, and therefore by which regular languages are manipulated, are not themselves regular. To see why this is so, suppose we have two regular expressions describing the first form and the root of the Arabic verb *to write*:

50. C V C V C
k (•* t)+ •* b

The intersection is the following regular expression:

51. k V t V b

The associations fixing the incidence of **k** with the first consonant slot, **t** with the third, and **b** with the final, are made by the intersection operation. The question arises as to how we can construct the associations if the same operation for Kornai's system is not regular. The operation we have applied here—intersection—cannot be performed by a regular transducer. This does not invalidate our claim to regularity. What is regular in our theory is each individual description and generalization about phonological data. That is, the descriptions we use are all regular descriptions of phonological objects.

What is not regular in one-level phonology is the relationship between different formats of the same description. There is no finite-state transducer that will form the product of two regular expressions. Multilevel analyses necessarily seek to capture relationships between different descriptions, and like the product operation, these relationships often cannot be captured by finite-state transducers.

6. Conclusions

The starting point of this paper was the distinction between descriptions and objects. Multidimensional phonological structures were taken to be descriptions of classes of phonetic objects, following Wheeler (1981), Bird and Klein (1990), Pierrehumbert (1990), Bird (1990), and Coleman (1992). Multiple tiers could be put together not by a clever encoding but by the simple operation of intersection, which corresponds to logical conjunction. Furthermore, this move of intensionalizing phonology enabled us to provide a straightforward formal basis for adding logical negation and disjunction to our representations.

One important consequence of this work is that there are now good prospects for the incorporation of nonlinear phonology into constraint-based grammar formalisms such as HPSG (Pollard and Sag 1987). Such a move gives rise to a novel view of the relationship between phonology and the other modules of grammar, as some initial investigation has already demonstrated (Bird 1992; Bird and Klein in press). Making surface generalizations the only goal of analysis makes the machine learning of analyses simpler (Ellison forthcoming). The automaton semantics for autosegmental representations and rules gives us a mechanical way of comparing the empirical claims made by a range of autosegmental and segmental accounts of natural language phenomena. Finally, to the extent that phonologists are becoming increasingly committed to a declarative, constraint-based view of their domain, we believe that the model proposed here is well suited to their computational needs.

Acknowledgments
This research is funded by the U.K. Science and Engineering Research Council, under grant GR/G-22084. *Computational Phonology:*

A Constraint-Based Approach. We are grateful to John Coleman, Mark Johnson, András Kornai, Ewan Klein, Henry Thompson, Markus Walther, Bruce Wiebe, and two anonymous reviewers for comments on this work. We are also grateful to the students who attended our course at the *Fifth European Summer School on Logic, Language and Information* (Lisbon, August 1993), and gave valuable feedback on this work. The authors take equal responsibility for the material presented here.

References

Antworth, E. (1990). *PC-KIMMO: A Two-Level Processor for Morphological Analysis*. Summer Institute of Linguistics.

Bear, J. (1986). "A morphological recognizer with syntactic and phonological rules." In *Proceedings, the 11th International Conference on Computational Linguistics*, 272–276.

Beesley, K. (1990). "Finite-state descriptions of Arabic morphology." In *Proceedings, Second Conference on Bilingual Computing in Arabic and English*.

Beesley, K.; Buckwater, T.; and Newton, S. (1989). "Two-level finite state analysis of Arabic." In *Proceedings, First Conference on Bilingual Computing in Arabic and English*.

Bird, S. (1990). *Constraint-based phonology*. Doctoral dissertation, University of Edinburgh.

Bird, S. (1992). "Finite-state phonology in HPSG." In *Proceedings, Fifteenth International Conference on Computational Linguistics*, 74–80.

Bird, S. (in press). *Computational Phonology: A Constraint-Based Approach*. Studies in Natural Language Processing. Cambridge University Press.

Bird, S., and Ellison, T. M. (1992). "One level phonology: autosegmental representations and rules as finite-state automata." RP 51, University of Edinburgh, Centre for Cognitive Science.

Bird, S., and Klein, E. (1990). "Phonological events." *Journal of Linguistics* 26, 33–56.

Bird, S., and Klein, E. (in press). "Phonological analysis in typed feature systems." *Computational Linguistics*, 20.

Bird, S., and Ladd, D. R. (1991). "Presenting autosegmental phonology." *Journal of Linguistics*, 27, 193–210.

Bloomfield, L. (1926). "A set of postulates for the science of language." *Language*, 2, 153–164. Reprinted in *Readings in Linguistics I: The Development of Descriptive Linguistics in America 1925–56*, edited by Martin Joos, 26–31.

Broe, M. (1993). *Specification Theory: The treatment of redundancy in generative phonology*. Doctoral dissertation, University of Edinburgh.

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

Clements, G. N. (1985). "The geometry of phonological features." In *Phonology Yearbook 2*, edited by C. Ewen and J. Anderson, 225–252. Cambridge University Press.

Clements, G. N. (1991). "Place of articulation in consonants and vowels: a unified theory." In *Working Papers of the Cornell Phonetics Laboratory, Number 5*, edited by G. N. Clements and E. Hume, 77–123. Phonetics Laboratory, Cornell University.

Clements, G. N., and Ford, K. C. (1979). "Kikuyu tone shift and its synchronic consequences." *Linguistic Inquiry*, 10, 179–210.

Coleman, J. S. (1991). *Phonological representations—their names, forms and powers*. Doctoral dissertation, University of York.

Coleman, J. S. (1992). "The phonetic interpretation of headed phonological structures containing overlapping constituents." *Phonology*, 9, 1–44.

Ellison, T. M. (1992). "Discovering vowel harmony." In *Background and Experiments in Machine Learning of Natural Language*, edited by W. Daelemans and D. Powers, 131–136. ITK.

Ellison, T. M. (1993). *The machine learning of phonological structure*. Doctoral dissertation, University of Western Australia.

Ellison, T. M. (forthcoming). "The iterative learning of phonological constraints." *Computational Linguistics*.

Goldsmith, J. A. (1976). *Autosegmental phonology*. Doctoral dissertation, Massachusetts Institute of Technology.

Goldsmith, J. (1982). "Accent systems." In *The Structure of Phonological Representations*, edited by H. van der Hulst and N. Smith, 47–63. Foris.

Goldsmith, J. (1991). "Phonology as an intelligent system." In *Bridges Between Psychology and Linguistics*, edited by D. J. Napoli and J. Kegl, 247–267. Lawrence Erlbaum.

Hopcroft, J., and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Johnson, C. D. (1972). *Formal Aspects of Phonological Description*. Mouton.

Karttunen, L. (1983). "KIMMO: A general morphological processor." *Texas Linguistic Forum*, 22, 163–186.

Kay, M. (1987). "Nonconcatenative

finite-state morphology." In *Proceedings, Third Meeting of the European Chapter of the Association for Computational Linguistics*, 2–10.

Kaye, J. (1989). *Phonology: A Cognitive View*. Erlbaum.

Kornai, A. (1991). *Formal Phonology*. Doctoral dissertation, Stanford University.

Kornai, A., and Kálman, L. (1988). "Hungarian sentence intonation." In *Autosegmental Studies on Pitch Accent*, edited by H. van der Hulst and N. Smith, 183–195. Foris.

Koskenniemi, K. (1983). *Two-level morphology: A general computational model for word-form recognition and production*. Doctoral dissertation, University of Helsinki.

Koskenniemi, K. (1985). "Compilation of automata from morphological two-level rules." In *Papers from the Fifth Scandinavian Conference of Computational Linguistics*, University of Helsinki, 143–149.

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

Leben, W. R. (1978). "The representation of tone." In *Tone—A Linguistic Survey*, edited by V. A. Fromkin, 177–219. Academic Press.

Mastroianni, M. (1993). "Attribute logic phonology." Technical Report CMU-LCL 93-4, Carnegie Mellon University.

McCarthy, J. (1981). "A prosodic theory of non-concatenative morphology." *Linguistic Inquiry*, 12, 373–418.

McCarthy, J. (1989). "Linear order in phonological representation." *Linguistic Inquiry*, 20(1), 71–99.

Narayanan, A., and Hashem, L. (1993). "On abstract finite-state morphology." In *Proceedings, Sixth Conference of the European Chapter of the Association for Computational Linguistics*, 297–304.

Partee, B.; ter Meulen, A.; and Wall, R. E. (1990). *Mathematical Methods in Linguistics*. Studies in Linguistics and Philosophy. Kluwer.

Pierrehumbert, J. (1990). "Phonological and phonetic representation." *Journal of Phonetics*, 18, 375–394.

Pollard, C., and Sag, I. (1987). *Information-Based Syntax and Semantics*, Volume 13 of *CSLI Lecture Notes*. Stanford: Center for the Study of Language and Information.

Pulleyblank, D. (1986). *Tone in Lexical Phonology*. Studies in Natural Language and Linguistic Theory. Reidel.

Pulman, S. G., and Hepple, M. R. (1993). "A feature-based formalism for two level phonology: a description and implementation." *Computer Speech and Language*, 7, 333–358.

Ritchie, G. D. (1992). "Languages generated by two-level morphological rules." *Computational Linguistics*, 18(1), 41–59.

Ritchie, G. D.; Russell, G. J.; Black, A. W.; and Pulman, S. G. (1992). *Computational Morphology: Practical Mechanisms for the English Lexicon*. MIT Press.

Russell, K. (1993). *A constraint-based approach to phonology*. Doctoral dissertation, University of Southern California.

Sagey, E. (1988). "On the ill-formedness of crossing association lines." *Linguistic Inquiry*, 19, 109–118.

Schiller, A., and Steffens, P. (1991). "Morphological processing in the two-level paradigm." In *Text Understanding in LILOG*, edited by O. Herzog and C.-R. Rollinger, 112–126. Springer-Verlag.

Scobbie, J. (1991). *Attribute-value phonology*. Doctoral dissertation, University of Edinburgh.

Sproat, R. (1992). *Morphology and Computation*. Natural Language Processing. MIT Press.

Touretzky, D. S., and Wheeler, D. W. (1990). "A computational basis for phonology." In *Advances in Neural Information Processing Systems 2: The Collected Papers of the 1989 IEEE Conference on Neural Information Processing Systems*, edited by D. S. Touretzky. Morgan Kaufmann.

Wheeler, D. (1981). *Aspects of a categorial theory of phonology*. Doctoral dissertation, Unversity of Massachusetts, Amherst, MA.

Wiebe, B. (1992). *Modelling autosegmental phonology with multi-tape finite state transducers*. Master's dissertation, Simon Fraser University.