Neural-Networks Models of Cognition
J.W. Donahoe and V. Packard Dorsel (Editors)

CHAPTER 25

SERIAL ORDER: A PARALLEL DISTRIBUTED PROCESSING APPROACH

Michael I. Jordan
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology

ABSTRACT

A theory of learned sequential behavior is presented, with a focus on coarticulatory phenomena in speech. The theory is implemented as a recurrent parallel distributed processing network that is trained via a generalized error-correcting algorithm. The basic idea underlying the theory is that both serial order and coarticulatory overlap can be represented in terms of relative levels of activation in a network if a clear distinction is made between the *state* of the network and the *output* of the network.

Introduction

Even the most cursory examination of human behavior reveals a variety of serially ordered action sequences. Our limb movements, our speech, and even our internal train of thought involve sequences of events that follow one another in time. We are capable of performing an enormous number of sequences, and we can perform the same actions in a variety of different contexts and orderings. Furthermore, most of the sequences that we perform were learned through experience.

A theory of serial order in behavior should clearly be able to account for these basic data. However, no such general theory has emerged, and an important reason for this is the failure of current formalisms to deal adequately with the parallel aspects of serially ordered behavior. We can tentatively distinguish two forms of parallelism. The first is parallelism that arises when actions in a sequence overlap in their execution. In speech research, such parallelism is referred to as *coarticulation* (Kent & Minifie, 1977; Moll & Daniloff, 1971; Öhman, 1966), and it greatly complicates the traditional description of sequential speech processes. The second form of parallelism occurs when two actions are required to be performed in parallel by the demands of the task or by implicit constraints. Such is the case, for example, in the *dual-task* paradigm, in which actions that have been learned separately must be performed together. This differs from the case of coarticulation, in which actions that are nominally separate in time are allowed to be performed in parallel. It is important to characterize both how such parallelism can arise within a sequential process and how it can be constrained so that unwanted parallel interactions are avoided.

In this paper, I present a theory of serial order that describes how sequences of actions might be learned and performed. The theory is embodied in the form of a parallel distributed processing network (Rumelhart & McClelland, 1986). Such networks are composed of a large number of simple processing units that are connected through weighted links. In various forms, such networks have been used as models of phenomena such as stereopsis (Marr & Poggio, 1976), word recognition (McClelland & Rumelhart, 1981), and reaching (Hinton, 1984). The success of these models has been due in large part to their high degree of parallelism, their ability to bring multiple interacting constraints to bear in solving complex problems, and their use of distributed representations. However, none of these properties seems particularly well suited to the problem of serial order. Indeed, a criticism of this class of models has been their inability to show interesting sequential behavior, whereas the more traditional symbolic approaches—typically by assuming a sequential processor as a primitive—deal with serial order in a much more straightforward manner. This criticism is challenged in this chapter, in the context of a theory of serial order that takes advantage of the underlying primitives provided by parallel distributed processing.

Serial Order

Many of the problems encountered in developing a parallel distributed processing approach to the serial-order problem were anticipated by Lashley (1951). Lashley pointed out the insufficiency of the *associative-chaining* solution to the serial-order problem. The associative-chaining solution assumes that serial ordering is encoded by directed links between control elements representing the actions to be ordered, and that the performance of a sequence involves following a path through the network of control elements. Lashley argued that this solution fails to allow different orderings of the same actions because there is no mechanism for specifying which link should be followed from an element having more than one outgoing link. He also argued that serial behavior shows anticipatory effects of future actions upon the current action, and that such context effects are not accounted for within the associationist framework.

Buffer approaches to serial order

Lashley's arguments have had an impact on those seeking to understand the role of feedback in a theory of motor behavior, but have been less influential on those interested in the structure of *motor programs*. This is in all likelihood due to the impact on theorists of the development of the digital computer, which made it possible to see how arbitrary sequential programs can be executed. Theories based explicitly on the computer metaphor have invoked the notion of a *buffer* which is loaded with the actions to be performed, and a program counter which steps through the buffer (Shaffer, 1976; Sternberg, Monsell, Knoll, & Wright, 1978). Despite the generality of such a theory, simple buffer theories are known to have several problems, including accounting for error patterns (Kent & Minifie, 1977; MacKay, 1981). It is also true that coarticulation is not well handled by buffer theories. One approach is to assume that buffer positions can interact with each other (Henke, 1966). However, this interaction, which must occur when successive actions are simultaneously present in the buffer, takes time, as does the process of reloading the buffer once a set of related actions has been executed. This approach implies the presence of delays at certain times in the production of long sequences, but such delays are not observed in fluent sequential behavior (cf. Shaffer & Hardwick, 1970). Another problem is that interactions between actions should depend on their relative positions in the buffer, not their absolute positions. For example, the interactions between the phonemes /*i*/ and /*n*/ should presumably be the same when saying "print" and "sprint." This would seem to imply the need for a complex mechanism whereby learned interactions can automatically generalize to all buffer positions. Such issues, which arise due to the explicit spatial representation of order in buffer theories, seem to be better handled within an associationist framework.

Associationist approach to serial order

Wickelgren (1969) revived the associationist approach by assuming that serial order was indeed encoded by directed links between control elements, but that the control elements were different for different orderings of the same actions. The control element for the action B in the sequence ABC can be represented by the form ${}_{A}B_{C}$ whereas the control element for B in the sequence CBA is represented as ${}_{C}B_{A}$. These control elements are distinct elements in the network, thus there is no problem with representing both the sequences ABC and CBA in the same network. In this account, actions are different in different contexts, not because they are executed in parallel, but because they are produced by different control elements.

Wickelgren's theory provides a solution to the problems posed by Lashley but it has several shortcomings. First, it requires a large number of elements, yet has difficulty with the pronunciation of words, such as "barnyard," that have repeated subsequences of length two or more (Wickelgren, 1969). Second, effects of context in speech have been shown to extend up to four or five phonemes forward in an utterance (Benguerel & Cowan, 1974). Extension of the theory to account for such effects would require an impossibly large number of control elements. Finally, note that there are only representations for *tokens* in the theory, and no representations for *types*. There is nothing in the theory to tie together the contextual variations of a given action. This means that there is no way to account for the linguistic and phonetic regularities that are observed when similar actions occur in similar contexts (Halwes & Jenkins, 1971).

Parallel-processing approaches to serial order

A different approach is to assume that actions are to some extent produced in parallel (Fowler, 1980; Rumelhart & Norman, 1982). The parallelism allows several control elements to influence behavior at a particular point in time, and therefore provides an account of coarticulatory effects, even though actions are represented in terms of context-free types. Rumelhart and Norman (1982) have shown that a model of typing incorporating parallelism can produce overlapping keystrokes much like those observed in transcription typing.

Allowing parallel activation of control elements accounts for context sensitivity; however, the problem of temporal ordering remains. Rumelhart and Norman achieved temporal ordering by assuming that elements suppress other elements through lateral inhibitory connections if they precede those elements in the sequence. This particular scheme is susceptible to Lashley's critique because all possible inhibitory connections must be present to allow the performance of the same elements in different orders, and a mechanism is needed for selecting the particular inhibitory connections used in the performance of a particular sequence. However, there are other ways of achieving the same effect that are not open to Lashley's critique (Grossberg, 1978; Grudin, 1981). Essentially, all of these schemes produce temporal order by inducing a graded activation pattern across the elements in the sequence, such that elements more distant in the future are activated less than earlier elements. Elements are assumed to influence behavior in proportion to their level of activation. Because the next action in the sequence is the most highly activated, it has the most influence on behavior. Once the activation of an element reaches a threshold, it is inhibited, allowing the performance of other items in the sequence.

A problem with these parallel-activation theories is that they have difficulty with sequences in which there are repeated occurrences of actions. In a pure type representation, there is simply no way to represent the repeated action. Rumelhart and Norman used a modified type representation in which they introduced special operators for doublings (e.g., AA) and alternations (e.g., ABA). However, they provided no general mechanism. For example, sequences such as ABCA invoked a parser to break the sequence into pieces, thus allowing no parallel influences across the break. This is not a satisfactory solution, in general, because data in speech show that coarticulatory influences can extend across sequences like ABCA (Benguerel & Cowan, 1974). Another possibility is to assume that repeated occurrences of actions are represented by separate control elements (representation by tokens). However, the combined effects of partially activated control elements will cause the first occurrence of a repeated action to move forward in time, whether or not this is actually desirable. Indeed, in a sequence such as ABBB, the B may overwhelm the A and be executed first. These problems are enhanced in featural representations of the kind that are often posited for actions (Grudin, 1983; Perkell, 1980; Rosenbaum, 1980) because the total activation from elements representing the repeated features will be greater than the activation levels for features that only occur once in the sequence, irrespective of the order of the features. Such problems arise because the single quantity of activation is being used to represent two distinct things: the parallel influences of actions and the temporal order of actions.

It is my view that many of these problems disappear when a clear distinction is made between the *state* of the system and the *output* of the system. Explicitly distinguishing between the state and the output means that the system has two activation vectors, which allows both temporal order and parallel influences to be represented in terms of activation. In the theory developed in this paper, the state and the output are assumed to be represented as patterns of activation on separate sets of processing units. These sets of units are linked by connections defining an *output function* for the system. Serial order is encoded both in the output function and in recurrent connections impinging on units representing the state; there is no attempt to encode order information in direct connections between the output units.

Coarticulation

In this section, I briefly introduce some of the parallel aspects of sequential behavior that have been considered important in the development of the current theory.

Several studies involving the recording of articulator trajectories have shown that speech gestures associated with distinct phonemes can occur in parallel. Moll and Daniloff (1971) showed that in an utterance such as "freon," the velar opening for the nasal */n/* can begin as early as the first vowel, thereby nasalizing the vowels. Benguerel and Cowan (1974) studied phrases such as "une sinistre structure," in which there is a string of the six consonants */strstr/* followed by the rounded vowel */y/*. They showed that lip-rounding for the */y/* can begin as early as the first */s/*. This is presumably allowable because the articulation of the consonants does not involve the lips.

These examples suggest that the speech system is able to take advantage of "free" articulators and use them in anticipating future actions. This results in parallel performance and allows speech to proceed faster and more smoothly than would otherwise be possible. Such parallelism clearly must be constrained by the abilities of the articulators. However, there are other constraints involved as well. In the case of "freon," for example, the velum is allowed to open during the production of the vowels because the language being spoken is English. In a language such as French, in which nasal vowels are different phonemically from non-nasal vowels, the velum would not be allowed to coarticulate with the vowels. Thus the articulatory control system cannot blindly anticipate articulations, but must be sensitive to phonemic distinctions in the language being spoken by only allowing certain coarticulations.

The situation is more complicated still if we note that constraints on parallelism may be specific to particular features. For example, in the case of */strstry/*, only the *rounding* of the */y/* can be anticipated. The *voicing* of the */y/* cannot be anticipated because that would change the phonemic identities of the consonants (for example, the */s/* would become a */z/*). Again, such knowledge cannot come from consideration of strategies of articulation, but must reflect higher-level phonemic constraints.

Thus, speech presents a difficult distributed-control problem in which constraints of various kinds are imposed on the particular patternings of parallelism and sequentiality that can be obtained in an utterance. What I wish to show in the remainder of this paper is how this problem can be approached with a theory based on parallel distributed processing networks.

A Theory of Serial Order

Let there be some sequence of *actions* $x_1, x_2, \ldots, x_r$, which is to be produced in the presence of a *plan* **p**. Each action is a vector in a parameter or feature space, and the plan can be treated as an action produced by a higher level of the system. The plan is assumed to remain constant during the production of the sequence, and serves primarily to designate the particular sequence that is to be performed.

In general, we would like the system to be able to produce many different sequences. Thus, different vectors **p** are assumed to be associated with different sequences of actions. A particular sequence is produced when a particular vector **p** is presented as input to the system. Note that, in principle, there need be no relationship between the form of plan vectors and the sequences that they evoke. Rather, a plan vector evokes a particular sequence because it was present as input to the system when the sequence was learned. Thus, plans may simply be arbitrary patterns of activation that serve to key particular sequences; they need not be scripts for the system to follow.

Actions are produced in a temporal context composed of actions nearby in time. This context entirely determines the desired action, in the sense that knowing the context makes it possible to specify what the current action should be. It is proposed that the system explicitly represents the temporal context of actions in the form of a state vector and chooses the current action by evaluating a function from states to actions. At each moment in time, an action is chosen based on the current state, and the state is then updated to allow the next action to be chosen. Serial order does not arise from direct connections between units representing the actions; rather, it arises from two functions that are evaluated at each time step: a function f which determines the output action x_n at time n,

$$x_n = f(s_n, \mathbf{p}) \tag{1}$$

and a function g which determines the state s_{n+1},

$$s_{n+1} = g(S_N, \mathbf{p}), \tag{2}$$

where both functions depend on the constant-plan vector as well as the current-state vector. Following the terminology of automata theory (Booth, 1967), f will be referred to as the *output* function, and g will be referred to as the *next-state* function. (From the definition, it can be seen that the plan **p** plays the role of the input symbol in a sequential machine. The use of the term "plan" is to emphasize the assumption that **p** remains constant during the production of the sequence. That is, we are not allowed to assume temporal order in the input to the system.)

Assumptions are made in the theory about the form of these functions. The output function f is assumed to arise through learned associations from state

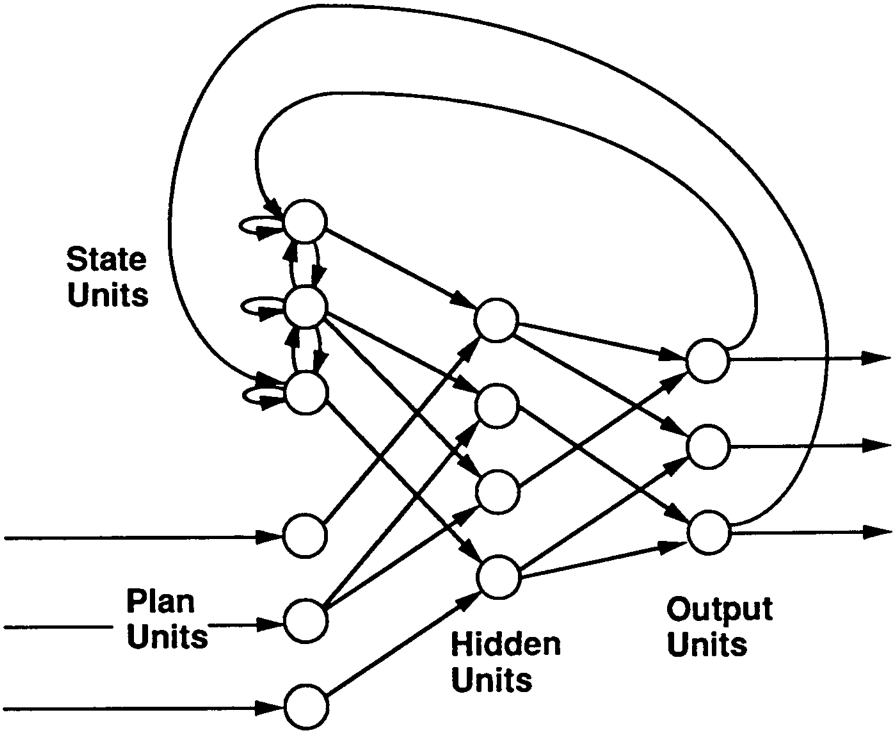

FIGURE 1. The processing units and basic interconnection scheme (not all connections are shown). The plan and state units together constitute the input units for the network.

and plan vectors to output vectors. These learned associations are assumed to generalize so that similar states and plans tend to lead to similar outputs. The major requirement for the next-state function g is that it have a continuity property: State vectors at nearby points in time are assumed to be similar. This requirement makes sense if the state is thought of as representing the temporal context of actions; intuitively, it seems appropriate that the temporal context should evolve continuously in time. Note that if the continuity property holds, then the generalizations made by the output function are such as to spread actions in time and, as learning proceeds, there is a tendency towards the increasing parallel execution of actions nearby in time. This process is discussed below in detail, where it is also shown how the generalizations leading to parallelism can be constrained.

A basic network architecture that embodies the theory is shown in **Figure 1**. The entities of the theory—plans, states, and outputs—are all assumed to be represented as distributed patterns of activation on three separate pools of processing units. The plan units and the state units together serve as the input units for a network that implements the output function f through weighted connections from the plan and state units to the output units. There are hidden units in the path from the plan and state units to the output units to allow for nonlinear output functions. Finally, the next-state function is implemented with recurrent connections from the state units to themselves and from the output units to the state units. This allows the current state to depend on the previous state and on the previous output (which is itself a function of the previous state and the plan).

In the proposed network, there is no explicit representation of temporal order and no explicit representation of action sequences. This is because there is only one set of output units for the network so that, at any point in time, only one output vector is present. Output vectors arise as a dynamic process, rather than being prepared in advance in a static buffer and then serially executed. Representing actions as distributed patterns on a common set of processing units has the virtue that partial activations blend together in a simple way to produce the output of the system.

Although it is possible that the next-state function as well as the output function arises through learning, this is not necessary for the system as a whole to be able to learn to produce sequences. Furthermore, given that the next-state function is set up in such a way that the continuity property holds, little is lost in the current framework if the recurrent connections necessary for the next-state function are taken as fixed and only the output function is learned. This latter approach is taken in the remainder of the chapter.

One choice of values for the fixed recurrent connections is based on the conception of the state as a temporal context. Consider the case of a sequence with a repeated subsequence or a pair of sequences with a common subsequence. It seems appropriate, given the positive transfer that can occur in such situations as well as the phenomena of capture errors (Norman, 1981), that the state should be similar during the performance of similar subsequences. This suggests defining the state in terms of the actions being produced. However, the representation must provide a sufficiently extensive temporal context that no ambiguities arise in cases involving repeated subsequences. If the state were to be defined as a function of the last *n* outputs, for example, then the system would be unable to perform sequences with repeated subsequences of length *n*, or to distinguish between pairs of sequences with a common subsequence of length *n*. To avoid such problems, the state can be defined as an exponentially weighted average of past outputs, so that the arbitrarily distant past has some representation in the state, albeit with ever-diminishing strength. This representation of the state is achieved if each output unit feeds back to a state unit with a weight of one, if each state unit feeds back to itself with a weight μ, and if the state units are linear. In this case, the state at time *n* is given by

$$S_N = \mu s_{n-1} + x_{n-1} \tag{3}$$

$$= \sum_{\tau=1}^{n-1} \mu^{\tau-1} x_{n-\tau} \tag{4}$$

The similarity between states depends on the particular actions that are added at each time step and on the value of μ. In general, with sufficiently large values of μ, the similarity extends forward and backward in time, growing weaker with increasing distance.

Learning and parallelism

In the network, learning is realized as an error-correcting process in which the weights of the network are incrementally adjusted based on the difference between the actual output of the network and a desired output. Essentially, the next-state function provides a time-varying state vector, and the error information drives changes in the mapping from this state vector and the plan vector to the output. The form that desired output vectors are assumed to take is a generalization of the approach used in traditional error-correction schemes (Rumelhart, Hinton, & Williams, 1986). Rather than assuming that a value is specified for each output unit, it is assumed that, in general, there are *constraints* specified on the values of the output units. Constraints may specify a range of values that an output unit may have, a particular value, or no value at all. This latter case is referred to as a "don't-care condition." It is also possible to consider constraints that are defined among output units; for example, the sum of the activations of a set of units might be required to take on a particular value. Constraints enter into the learning process in the following way: If the activation of an output unit fits the constraints on that unit, then no error corrections are instigated from that unit. If, however, a constraint is not met, then the error is defined as a proportion of the degree to which that constraint is not met, and this error is used in changing system parameters towards a configuration in which the constraint is met.

In many realistic sequence-learning problems, it would seem that desired outputs cannot be assumed to be directly available at the output units of the network. For example, in the case of speech production, the information provided to the learner is auditory or perceptual, whereas desired output information for the production module must be specified in terms of articulator motion. A related problem is that target information may be delayed in time relative to performance. Such problems of a "distal teacher" have been addressed in recent work that shows how the constraints may themselves be learned (Jordan & Rumelhart, 1992). The constraints are implemented in an auxiliary network that models the mapping from the network outputs to the distal results. Once the model is learned, backpropagation through the model converts distal error vectors into error vectors for the output units. For example, if the auxiliary network models the mapping from articulatory events to auditory events, then backpropagation can be used to convert auditory errors backward into articulatory errors. The error vectors that are computed by this process can be thought of as providing target outputs for the underlying sequential network. Thus, for current purposes, we can make the simplifying assumption that desired outputs are provided directly by an external agent. There is a caveat, however: When the auxiliary network models a many-to-one function, then the error vectors computed by backpropagation implicitly specify a region in output space, rather than a point. Of course, it is precisely this underspecification that is of interest, because it allows actions in a sequence to have an effect on one another. Here, I use don't-care conditions in the specification of desired output vectors to allow consideration of a particularly simple case: regions that are rectangular and parallel to the axes of the output space. For further discussion of the general case, see Jordan (1990).

Consider first the case in which desired output vectors specify values for only a single output unit. Suppose that a network with three output units is learning the sequence

$$\begin{bmatrix} .9 \\ * \\ * \end{bmatrix}, \begin{bmatrix} * \\ .9 \\ * \end{bmatrix}, \begin{bmatrix} * \\ * \\ .9 \end{bmatrix}.$$

The network is essentially being instructed to activate its output units in a particular order, and this case can be thought of as involving local representations for actions. At each time step, errors are propagated from only a single output unit, so that activation of that unit becomes associated to the current state s_i. Associations are learned from s_1 to activation of the first output unit, from s_2 to activation of the second output unit, and from s_3 to activation of the third output unit. These associations also generalize so that, for example, s_1 tends to produce partial activations of the second and third output units. This occurs because s_1 is similar to s_2 and s_3, and—by the assumption of continuity of the next-state function—similar inputs produce similar outputs in these networks. After learning, the network will likely produce a sequence such as

$$\begin{bmatrix} .9 \\ .7 \\ .5 \end{bmatrix}, \quad \begin{bmatrix} .7 \\ .9 \\ .7 \end{bmatrix}, \quad \begin{bmatrix} .5 \\ .7 \\ .9 \end{bmatrix},$$

where at each time step, there are parallel activations of all output units. If the network is driving a set of articulators that must travel a certain distance, or have a certain inertia, then it will be possible to go faster with these parallel control signals than with signals where only one output unit can be active at a time.

The foregoing example is simply the least constrained case and further constraints can be added. Suppose, for example, that the second output unit is not allowed to be active during the first action. This can be encoded in the target vector for the first action so that the network is instructed to learn the sequence

$$\begin{bmatrix} .9 \\ .1 \\ * \end{bmatrix}, \quad \begin{bmatrix} * \\ .9 \\ * \end{bmatrix}, \quad \begin{bmatrix} * \\ * \\ .9 \end{bmatrix}.$$

After learning, the output sequence will likely be as follows:

$$\begin{bmatrix} .9 \\ .1 \\ .5 \end{bmatrix}, \quad \begin{bmatrix} .7 \\ .9 \\ .7 \end{bmatrix}, \quad \begin{bmatrix} .5 \\ .6 \\ .9 \end{bmatrix},$$

where the added constraint is now met. In this example, the network must block the generalization that is made from s_2 to s_1.

As further constraints are added, and fewer generalizations across nearby states are allowed, performance becomes less parallel. Minimal parallelism will arise when neighboring actions specify conflicting values on all output units, in which case the performance will be strictly sequential. Maximal parallelism should be expected when neighboring actions specify values on nonoverlapping sets of output units. Note that there is no need to invoke a special process to introduce parallelism into the system. Parallelism arises from the ability of the system to generalize, and is a manifestation of the normal functioning of the system. Indeed, in most cases, it will be more difficult for the system to learn in the strictly sequential case when there are more constraints imposed on the system.

Serial order

Before turning to a more detailed discussion of coarticulation, it is worth considering how the current theory fares with respect to some of the general requirements of a theory of serial order. It should be clear that the theory can deal with the problem of converting a static input into a time-varying output, given that the state changes over time, and given that an appropriate output function can be constructed. Different orderings of the same actions can be achieved, both because the state trajectories may differ between the sequences and because the output function depends on the plan, and the plan can distinguish the different orderings. The theory has no problem with repeated actions; the existence of repeated actions simply indicates that the output function is not one-to-one, but that two or more state, plan pairs can map to the same output vector. Finally, sequences such as ABAC, which cause problems for an associative-chaining theory because of the transitions to distinct actions after a repeated action, are possible because the state after the first A is not the same as the state after the second A.

The theory is able, in principle, to account for a variety of regularities that occur within and between sequences. This is because outputs and states are represented as types; that is, there is only one set of output units and one set of state units. The same weights underlie the activation of actions, in whatever position in the sequence, and in whatever sequence. Thus, particular weights underlie the regularities observed for similar actions in similar contexts. For example, the fact in English that voiceless stops are aspirated following */s/* (e.g., */spIn/* is pronounced [*sbIn*]), would be encoded by inhibitory connections from state units encoding the recent occurrence of a voiceless alveolar fricative to output units controlling glottal and labial movements. In the sequential-network architecture, this encoding allows the allophonic regularity to generalize immediately in contexts other than the initial portion of the word. Such a sensitivity to relative position, rather than absolute position, is difficult to obtain in architectures using spatial buffers (Sejnowski & Rosenberg, 1986), and problematic to obtain (in full generality) in schemes using context-sensitive allophones (Rumelhart & McClelland, 1986; Wickelgren, 1969).

One of the more important tests of a theory of serial order is that it account for interactions both forward and backward in time. In the current theory, time is represented implicitly by the configuration of the state vector. Interactions in time are due to the similarity of the state vector at nearby points in time. There is no time arrow associated with this similarity, thus, forward and backward interactions are equally possible.

Limitations on the structure of the functions f and g will lead to some sequences being more difficult to learn and perform than others. For example, the temporal context cannot extend indefinitely far in time; thus, the repetition of lengthy subsequences that make transitions to different actions can be difficult to learn and perform. Also, similarity between action transitions in different plans can cause interference, as can similarity between plan representations. The interference can lead to errors and to the learning of one sequence causing negative transfer on another sequence. Such interactions can also have a positive side, of course, in the form of positive transfer.

Dynamic properties of the networks

When a network learns to perform a sequence, it essentially learns to follow a trajectory through a state space. The state space consists of the ensemble of possible vectors of activation of the output units. An important fact about the learned trajectories is that they tend to influence points nearby in the state space. Indeed, the learned trajectories tend to be *attractors*.

Consider, for example, a network taught to perform the cyclic sequence

$$\begin{bmatrix} .25 \\ .25 \end{bmatrix}, \begin{bmatrix} .75 \\ .25 \end{bmatrix}, \begin{bmatrix} .75 \\ .75 \end{bmatrix}, \begin{bmatrix} .25 \\ .75 \end{bmatrix}, \begin{bmatrix} .25 \\ .25 \end{bmatrix}.$$

The trajectory of the network is on the four corners of a square in the first quadrant of the plane. The trajectory will repeatedly move around this square if the initial vector of activations of the output units is one of the corners of the square. It is also possible to set the initial activations of the output units to other values, thereby starting the network at points in the space other than the four corners of the square. **Figure 2** (left panel) shows the results of a simulation experiment in which the network was started at the point (.4,.4). As can be seen, the trajectory spirals outward and begins to approximate the square more and more closely. When the network is started at a point outside of the square, the trajectory is found to spiral inward towards the square. A sample trajectory starting from the point (.05,.05) is shown in the right panel of **Figure 2**. When the network was initialized at each of 100 points in the state space, it was found that all trajectories reached the square in the limit, demonstrating that the square is a *periodic attractor*. Note that trajectories starting inside the square approach the limit cycle less rapidly than do trajectories starting outside the square. At a point inside the square, the trajectory is subject to influences associated with all four corners, and these influences are in conflicting directions and therefore tend to cancel one another. At a point outside the square, however, only a pair of adjacent corners tend to influence the trajectory, and adjacent influences do not conflict in this example.

The dynamics exhibited by the networks described above has several useful properties. The system tends to be noise-resistant, because perturbed trajectories return to the attractor trajectory. The system is also relatively insensitive to initial conditions. Finally, the learning of a particular trajectory automatically generalizes to nearby trajectories, which is what is desired in many situations. The relevance of these properties to motor control has been recognized by several authors (Kelso, Saltzman, & Tuller, 1987; Saltzman & Kelso, 1987). I wish to suggest that such dynamics may also characterize the higher-level dynamic system that is responsible for serial ordering.

Application of the theory to coarticulation

The theory presented in this paper involves a dynamic system that is constrained through a learning process to follow particular trajectories. The learning process relies on constraints on the output of the system. These constraints implicitly define regions in output space through which trajectories must pass, and thereby delimit the possible range of effects of temporal context.

In the case of speech, the form of the constraints on articulation depends on inter-articulator organization, both kinematic and dynamic, and on the function

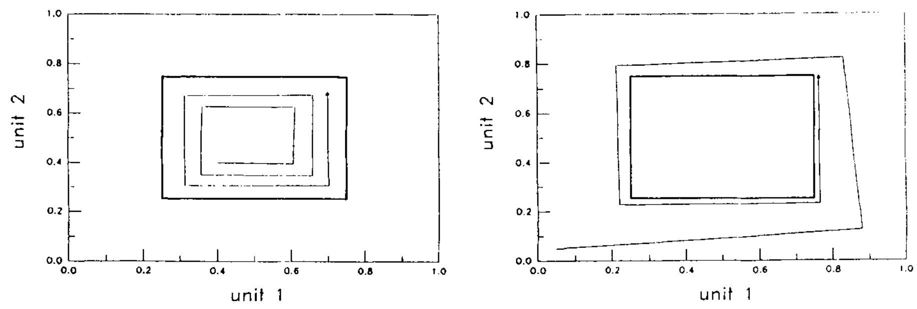

FIGURE 2. Two examples of the activations of the two output units plotted with time as a parameter. In each case, the square is the trajectory that the network learned. Left panel: The spiral trajectory is the path that the network followed when started at the point (.4,.4). Right panel: The spiral trajectory is the path that the network followed when started at the point (.05,.05).

that relates articulatory events to perceptual events. This latter function includes at least two kinds of mappings—one that relates articulator motion to pre-categorical auditory representations, and one that relates pre-categorical representations to post-categorical representations. After preliminary learning, both of these mappings can be assumed to be represented internally and thereby available to compute articulatory constraints from perceptual data as discussed in Jordan (1990). The salient characteristic of both of these mappings is that they are many-to-one (cf. Atal, Chang, Mathews, & Tukey, 1978). Thus, during imitative learning, the error vectors that are computed from the back-propagation of perceptual information implicitly specify *regions* of articulatory space rather than points. As described previously, the underlying dynamic system will form trajectories that pass smoothly through these regions. This yields contextually dependent variants of a given articulatory equivalence class. In summary, coarticulation is hypothesized to be a form of smoothness in articulatory space that is subject to perceptual constraints.

The perceptual information that provides target vectors for imitative learning may be either pre-categorical or post-categorical. Clearly, children's ability to acquire accent and other non-distinctive aspects of speech suggests that learning must be at least partially based on pre-categorical target information. It is tempting to hypothesize that the locus of target information evolves as post-categorical representations are formed over the course of development: Using a post-categorical target specifies a larger region of articulatory space, and therefore allows more flexibility in the choice of an articulatory trajectory. Of course, this flexibility is obtained with a corresponding loss in the ability to acquire articulation that reflects pre-categorical details.

In this section, I present some simple simulations of a system learning phonetic sequences. It should be emphasized that I am not proposing a realistic model of speech production in this section. A major simplification is that I have defined desired output vectors directly in articulatory terms using target values and don't-care conditions. This representation ignores the problem of converting perceptual information into articulatory information as well as the effects of articulatory dynamics. (Both of these issues can be addressed, however, within the framework of the forward-modeling approach; see Jordan & Rumelhart, 1992.) Nonetheless, the simulations are useful in elucidating the network algorithms hypothesized to underlie coarticulation. Also, they allow some qualitative predictions to be made.

The problem of serial ordering in speech is typically treated in discrete terms, and the relation between discrete higher-level processes and continuous lower-level articulatory processes has provoked much debate in the literature on speech production (Fowler, 1980; Hammarberg, 1982; Perkell, 1980). In the current theory, however, such issues are not particularly problematic because the entire system can be thought of as operating in continuous time. It is consistent with the current theory to assume that the defining state equations are simply a discrete version of a continuous-time dynamic system. In the continuous case, learning involves imposing constraints intermittently on the system at various points in time. In geometric terms, constraints appear as regions through which continuous-network trajectories must pass, with trajectories between regions unconstrained. To approximate the continuous system in the simulation, I have inserted several time steps between steps at which constraints are imposed. During these intermediate time steps, the network is free running (these intermediate steps can be thought of as having don't-care conditions on all of the output units). By conducting the simulation in this manner, it is possible to demonstrate the differences between the current approach and an assimilatory model in which different allophones are produced at each time step and interactions must begin and end at allophonic boundaries (cf. Fowler, 1980).

Feature	*i*	*s*	*t*	*r*	*s*	*t*	*r*	*y*
voice	8	1	1	*	1	1	*	8
place	7	9	9	2	9	9	2	7
sonorant	8	2	1	5	2	1	5	8
sibilant	1	9	2	4	9	2	4	1
nasal	*	*	1	*	*	1	*	*
height	9	9	9	9	9	9	9	9
back	1	*	*	2	*	*	2	1
round	1	*	*	*	*	*	*	9

TABLE 1. Target vectors for the string */istrstry/*.

Simulation experiments

For the purposes of describing the simulations section, I use the term "phoneme" to refer to a vector of target values and don't-care conditions. Representations for the phonemes were adapted from a list of real-valued features proposed by Ladefoged (1982). Eight features were selected that provided adequate discriminations between the particular phonemes used in the simulations. The feature values were all between 0.1 and 0.9. Choices for don't-care conditions were based on known allophonic variations (for example, the rounding for the French */s/* was taken to be a don't-care condition, because it is possible to have a rounded or an unrounded */s/*).

The network used in the simulations had 8 output units, 10 hidden units, 6 plan units, and 8 state units. The state units had recurrent connections onto themselves with weights of $\mu = 0.5$.

The procedure used in the simulation was essentially that of the preceding section, with the following modification. During learning trials, target vectors (i.e., phonemes) were presented to the network every fourth time step. Learning occurred only on these time steps. During the intermediate three time steps, the units were updated normally with no learning occurring.

In the first experiment, the network was taught to perform the sequence "sinistre structure." The phonemes that were used are shown in **Table 1** for the embedded sequence */istrstry/* only. The learning process involved repeated trials in which the phonemes in the sequence were used as target vectors for the network. The plan was a particular constant vector whose composition is irrelevant here because the network learned only this one sequence. The results for the embedded sequence */istrstry/* are shown in **Figure 3**, which displays the output trajectories actually produced by the network once the sequence was learned to criterion. The network learned to produce the specified values, as can be seen by comparing the values produced at every fourth time step with the values in the table. The network also produced values for the don't-care conditions and for unconstrained parts of the trajectories. In particular, the value of .9 for the rounding feature of the rounded vowel */y/* was anticipated as early as the third time step. In a control experiment, the sequence "sinistre stricture," in which the same consonant sequence is followed by the unrounded vowel */i/*, was taught to the network. As shown in **Figure 4**, there was now no rounding during the entire utterance. These results parallel the data obtained by Benguerel and Cowan (1974).

In a third experiment, the network learned the sequence "freon," where the feature of interest was the nasal feature associated with the terminal */n/*. In the phoneme vectors, the */f/* was specified as 0.1 for the nasal feature, the */n/* was specified as 0.9, and the intervening three phonemes had don't-care values for the nasal feature. Thus, this experiment is analogous to the previous experiment, with the interest in the anticipation of the nasal feature rather than the rounding feature. The results are shown in **Figure 5** in terms of the activation of the nasal feature at every fourth time step. As in the data of Moll and Daniloff (1971), there was substantial anticipation of the nasal value of */n/* before and during the two vowels. Note that a steeper dropoff in the amount of anticipation occurred in this sequence than in the sequence */istrstry/*. An investigation of the weights learned during these sequences revealed that the extensive coarticulation in the latter sequence arose from the repetition of phonemes. The rounding of */y/* was produced in a temporal context in which

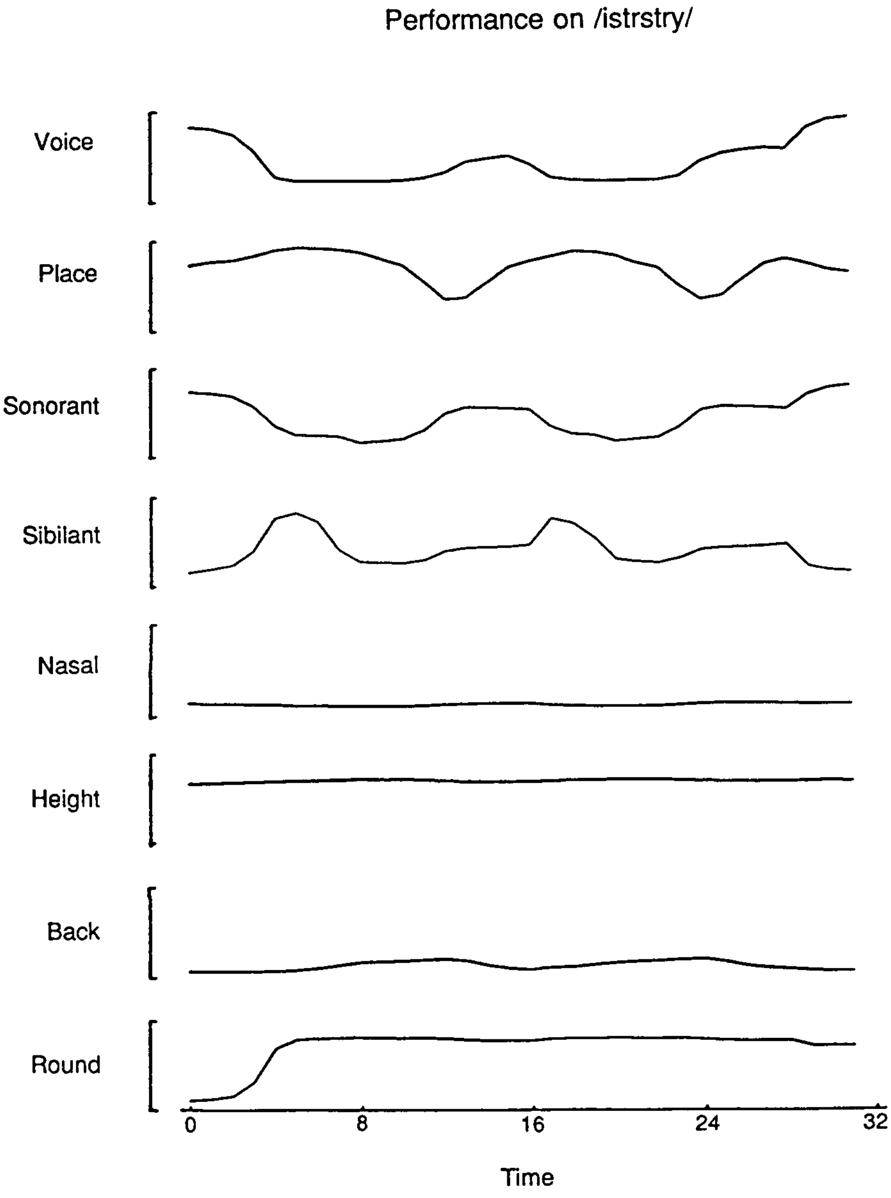

FIGURE 3. Output trajectories for the sequence /*istrstry*/.

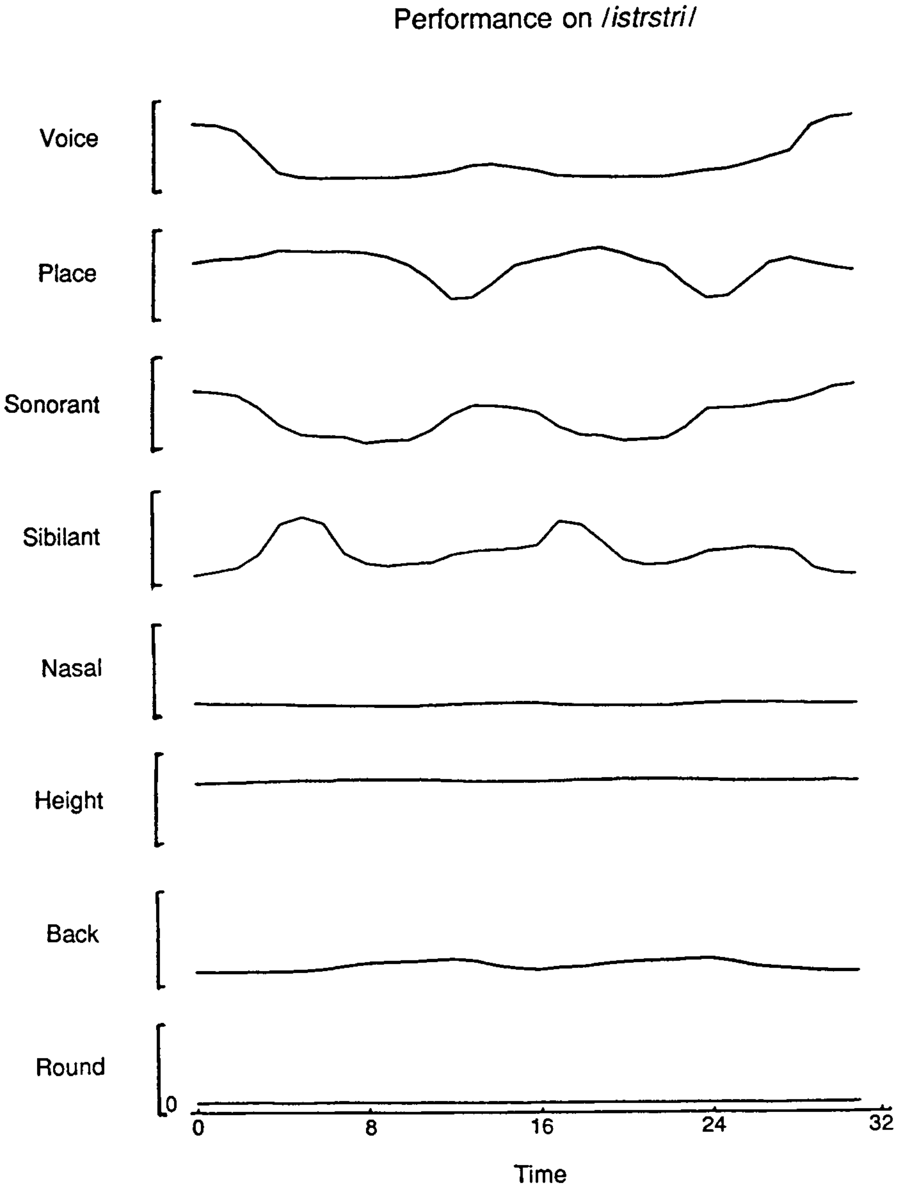

FIGURE 4. Output trajectories for the sequence */istrstri/*.

/*str*/ was the preceding subsequence. A very similar context occurred after the first /*r*/, thus, there was necessarily coarticulation into the first repetition of /*str*/. These considerations suggest that, in general, more forward coarticulation should occur over strings that have homogeneous phonemic structure than over strings with heterogeneous phonemes.

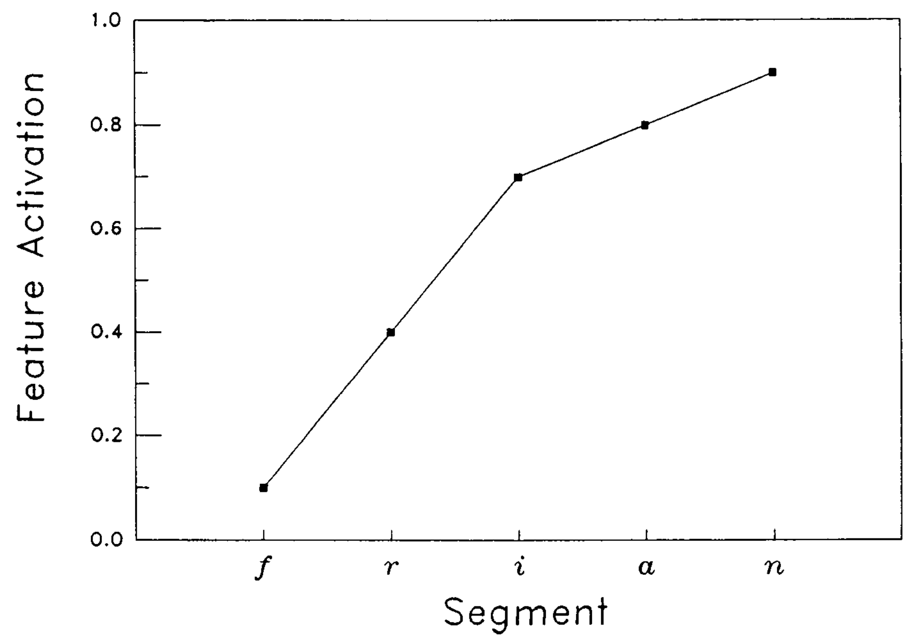

FIGURE 5. Activation of the nasal feature at every fourth time step during performance of the word "freon."

Another interesting aspect of the simulated coarticulation can be seen by considering the voicing feature in **Figure 5**. This feature is unspecified for the phoneme /*r*/ (the /*r*/ in French can be voiced or unvoiced depending on the context; compare "rouge" and "lettre"), but is specified as a 0.1 for the directly adjacent features /*t*/ and /*s*/. Nevertheless, the first /*r*/ receives a small amount of voicing, which comes from the positive value of voicing for the nearby, but not adjacent, phonemes /*i*/ and /*y*/. This result emphasizes the underlying mechanism of activation of the output units: Units are activated to the extent that the current state is similar to the state in which they were learned. This means that units with don't-care conditions take on values that are, in general, a compromise involving the values of several nearby phonemes, and not merely the nearest specified value. Typically, however, the nearest phoneme has the most influence.

These considerations suggest that the amount of forward coarticulation should depend not only on the preceding phonemes, but also on the following phoneme. If the phoneme following /y/ is unrounded, for example, then rounding of the /y/ should be anticipated less than when the following phoneme is rounded or unspecified on the rounding feature (as in the example of "structure"). This prediction was borne out in simulation. The French pseudowords "virtuo," "virtui," and "virtud," in which the rounded phoneme /y/ is followed by the rounded phoneme /o/, the unrounded phoneme /i/, or the "don't-care" phoneme /d/, were taught to the network. The results are shown in **Figure 6** in terms of the activation of the rounding feature at successive points in time. The figure shows that forward coarticulation in the network clearly depends on the following context.

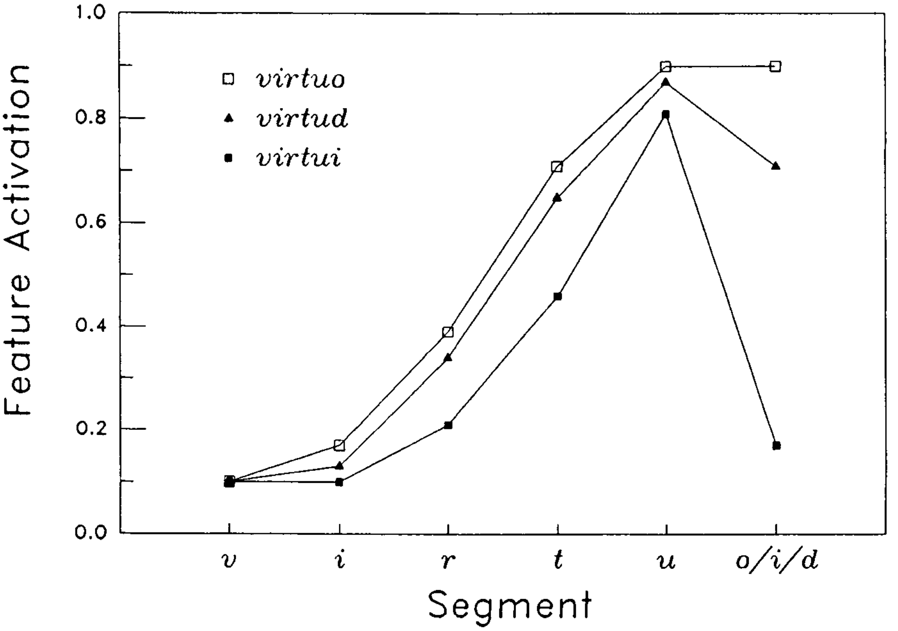

FIGURE 6. Activation of the rounding feature at every fourth time step during performance of three French pseudowords.

Discussion

In their review on coarticulation, Kent and Minifie (1977) distinguish between submovements in an articulatory sequence that have "immediate successional impact," that is, those that "must follow one another in a prescribed sequence," and submovements without immediate successional impact, that are "accommodated within the sequential pattern defined by the locally critical articulatory transitions." The model presented in this section obeys this distinction, where constraints specify the locally critical articulatory transitions. The model also provides a mechanism for the process of "accommodation," by which features without immediate successional impact can be integrated into the articulatory program.

It is worthwhile to compare the current simulation to a feature-spreading model such as that proposed by Henke (1966). Henke's model is essentially a buffer model, in which positions in the buffer are loaded with the phonemes to be produced. Phonemes are lists of trinary features, each of which can have the value +, -, or 0. When a buffer position is to be executed, features having value 0 are filled in by an operator that serially inspects "future" buffer positions until a plus or a minus is found. Once all features are filled in, the allophonic variation thus created can be executed. Although this model is similar to the current simulations in the sense that both rely on context-independent representations of phonemes that specify dimensions along which the phonemes can be altered, there are important differences.

From a conceptual point of view, the underlying mechanisms that determine output values are quite different and have different empirical consequences. In the current approach, parallel performance arises automatically, without the need for a special process to program in the parallelism. This occurs because the current state is similar to the state in which nearby phonemes were learned, and similar states tend to produce similar activations of the output units. There is therefore no implication that features can spread indefinitely in time, which is true of a strict interpretation of Henke's model (Gelfer, Harris, & Hilt, 1981). Rather, the spread of a feature in time diminishes due to the dropoff in similarity of the state. For related reasons, there is no implication that feature vectors change discretely in time. As the state evolves continuously in time, the components of the output vector also evolve continuously in time, with no necessary coherence between anticipated or perseverated features and adjacent segments (cf. Fowler, 1980). Indeed, there is really no notion of a segment in the output of the network. Also, whereas Henke's model is an assimilatory model of coarticulation, the current model is best thought of as a model of parallelism in speech production. As shown in the simulations, the parallel model predicts nonadjacent interactions: For example, the amount of forward coarticulation of a feature in a phoneme depends on what follows the phoneme. Although an assimilatory model could be constructed to mimic such behavior, it would seem better accounted for within the parallel approach. However, I know of no empirical evidence relevant to deciding this issue. (The following experiment would constitute a critical test. Consider forward vowel-to-vowel coarticulation, such as the raising of */a/* when it is followed by */i/* in the sequence */papi/* (Manuel & Krakow, 1984). When this sequence is followed by */e/* (e.g., in the sequence */papipe/*), the */e/* acts to lower the */i/*. The question is what happens to the */a/* in */papi/* vs. */papipe/*. Under an assimilation hypothesis, the */a/* should be lower in the latter case because the source for its raising (the */i/*) has been lowered. Under the hypothesis of parallelism, on the other hand, the */a/* should be at least as high in */papipe/* as in */papi/*, because both the */e/* and the */i/* act to raise */a/*.) Finally, it should be noted that in the current model, utterances are not explicitly represented (i.e., in a buffer) before being produced. Rather, the process is truly dynamic; utterances are implicit in the weights of the network, and become explicit only as the network evolves in time.

The simulations presented above relied only on the simplest constraints on the output units. However, much could be gained by considering more complex constraints such as inequality constraints, range constraints, or constraints between units. For example, certain low-level effects of context, such as the dentalization of the */d/* in "width," are often treated as phonological in origin, rather than resulting from coarticulation. This is presumably because the place of articulation jumps discretely to dental, rather than moving somewhere between alveolar and dental. In the current model, however, the */d/* could be represented as having a range constraint on the place of constriction feature (i.e., a constraint that the place be between a pair of values). The actual value chosen for the place feature will be dependent on the neighboring context, and a context such as the dental fricative could well drive this value against a boundary of the range constraint. Similarly, constraints between units can determine which articulatory configuration is chosen out of several possibilities. For example, if the sum of the activations of three output units must be a particular value, then it is possible to trade off the activations among the units if particular units are further constrained by the neighboring context. Finally, the general case of learned, nonlinear constraints allows modeling of the role of the nonlinear mapping from articulation to acoustics in determining the way in which articulatory components trade off (Jordan, 1990).

There are two possible versions of a parallel model of coarticulation. The first assumes that parallelism is feature-specific, that is, that particular features of a phoneme can be anticipated or perseverated. This approach is consistent with the distinction of Kent and Minifie (1977) discussed above, and is the approach that I have emphasized. However, it is also possible to assume that all of the components of a phoneme must be activated together. This is the approach favored by Fowler (1980), who claims that coarticulation results from the coproduction of "canonical forms." In the current framework, such phoneme-specific parallelism occurs when phonemes specify constraints on nonoverlapping sets of output units. In the limiting case, each phoneme can constrain a unique output unit, in which case the partial activations of output units lead to the partial production of entire phonemes rather than specific features. It is still possible to represent phonemes by features, but this must be done at a lower level in the system, below the level at which parallelism arises.

However, it would appear that feature-specific parallelism is necessary. For example, in the production of a sequence of vowels followed by an */n/*, it would seem important that only the velar movement associated with the nasal be anticipated, and not the alveolar tongue position. There is some evidence for this in the data of Kent, Carney, and Severeid (1974). In recordings of the articulatory movements during the utterance "contract," they found that the movement towards the alveolar tongue position for the */n/* began 120 milliseconds *after* the onset of velar lowering for the */n/*. This suggests that the features of the */n/* are not being controlled synchronously.

To summarize, the current proposal is that coarticulation results from the similarity structure of the state at nearby points in time. The dropoff in similarity of the state defines the zone in which the features of a phoneme can possibly be present in the output. Within this zone, the pattern of coarticulation that is obtained depends on the constraints that are imposed by the features corresponding to nearby phonemes.

Conclusions

The current theory provides an alternative to the traditional motor-program approach to the serial-order problem. The traditional approach, based on the von Neumann conception of a stored program, assumes that motor actions are instructions that are assembled into a structure that is then scanned by a sequential processor. The parallelism and interactiveness of real behavior prove burdensome to such an approach, and typically, extra mechanisms must be invoked. In the current approach, on the other hand, parallelism is a primitive, arising directly from the continuity of the mappings defining the system. Strictly sequential performance is simply the limiting, most highly constrained case.

This chapter has concentrated on only certain aspects of the serial-order problem, namely those involving learning and coarticulation. Jordan (1986) discusses other issues, including rate, errors, hierarchies, and dual-task parallelism.

The concept of state is central to the current theory. Time is represented implicitly by the configuration of the state vector, and it is the assumption of a continuously varying state that relates nearby moments in time and provides a natural way for behavior to be parallel and interactive locally while still broadly sequential. The similarity structure of the underlying state provides a theoretical point of convergence for many kinds of behavioral data. The pattern of coarticulation depends on this similarity structure, errors are more likely when discriminations must be made between similar states, dual-task interference is a function of similarity, and learning is faster when similar actions are associated to similar states. Thus, if the theory is to prove useful, elucidation of the similarity structure of the states underlying sequential behavior becomes an overriding theoretical and empirical concern.

ACKNOWLEDGMENTS

This chapter is an abridged version of Jordan (1986). My sincere thanks to Donald Norman and David Rumelhart for their support of many years. I also wish to acknowledge the help of Eileen Conway, Kathy Farrelly, Jonathan Grudin, Bernhard Keller, Michael Mozer, David Navon, and Stanley Parkinson. This research was supported by Contract N00014-85-K-0450, NR 667-548 with the Personnel and Training Research Programs of the Office of Naval Research.