CHAPTER 4

PDP Models and General Issues in Cognitive Science

D. E. RUMELHART and J. L. MCCLELLAND

We are naturally optimistic about parallel distributed processing as a valuable framework for creating cognitive models. This does not mean, however, that there are no tough problems to be solved. Indeed, we have spent much of our effort convincing ourselves that PDP models could form a reasonable basis for modeling cognitive processes in general. In this chapter we shall address some of the objections that we and others have raised to the work and sketch our answers to these objections. However, we should like to say at the outset that we do not believe that any such general considerations as those discussed here will, in the end, bear much weight. The real proof is in the pudding. If PDP models are a valuable way to proceed, their usefulness will be proved in the added insights they bring to the particular substantive areas in which they are applied. The models we describe in later chapters are largely intended to constitute the beginnings of such a proof.

Many of the questions and issues raised below are addressed by material described in detail in other chapters in the book. For this reason, much of our present discussion is in the form of pointers to the relevant discussions. In this sense, this chapter serves not only as a discussion of our approach but as an overview of the issues and topics that are addressed in the chapters that follow.

SOME OBJECTIONS TO THE PDP APPROACH

PDP Models Are Too Weak

The one-layer perceptron. The most commonly heard objection to PDP models is a variant of the claim that PDP models cannot perform any interesting computations. One variant goes like this: "These PDP models sound a lot like perceptrons to me. Didn't Minsky and Papert show that perceptron-like models couldn't do anything interesting?" This comment represents a misunderstanding of what Minsky and Papert (1969) have actually shown. A brief sketch of the context in which Minsky and Papert wrote will help clarify the situation. (See Chapter 5 for a somewhat fuller account of this history.)

In the late 1950s and early 1960s there was a great deal of effort in the development of self-organizing networks and similar PDP-like computational devices. The best known of these was the *perceptron* developed by Frank Rosenblatt (see, for example, Rosenblatt, 1962). Rosenblatt was very enthusiastic about the perceptron and hopeful that it could serve as the basis both of artificial intelligence and the modeling of the brain. Minsky and Papert, who favored a *serial symbol processing* approach to artificial intelligence, undertook a very careful mathematical analysis of the perceptron in their 1969 book entitled, simply, *Perceptrons.*

The perceptron Minsky and Papert analyzed most closely is illustrated in Figure 1. Such machines consist of what is generally called a *retina*, an array of binary inputs sometimes taken to be arranged in a two-dimensional spatial layout; a set of *predicates*, a set of binary threshold units with fixed connections to a subset of units in the retina such that each predicate computes some local function over the subset of units to which it is connected; and one or more decision units, with modifiable connections to the predicates. This machine has only one layer of modifiable connections; for this reason we will call it a *one-layer* perceptron.

Minsky and Papert set out to show which functions can and cannot be computed by this class of machines. They demonstrated, in particular, that such perceptrons are unable to calculate such mathematical functions as parity (whether an odd or even number of points are on in the retina) or the topological function of connectedness (whether all points that are on are connected to all other points that are on either directly or via other points that are also on) without making use of absurdly large numbers of predicates. The analysis is extremely elegant and demonstrates the importance of a mathematical approach to analyzing computational systems.

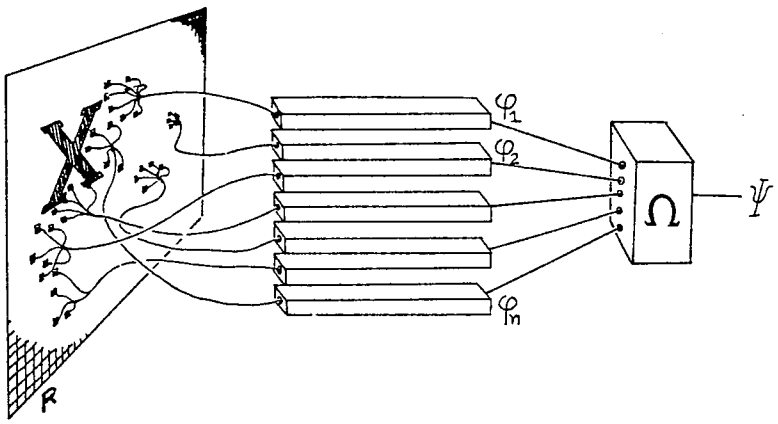

FIGURE 1. The one-layer perceptron analyzed by Minsky and Papert. (From *Perceptrons* by M. L. Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press. Reprinted by permission.)

Minsky and Papert's analysis of the limitations of the one-layer perceptron, coupled with some of the early successes of the symbolic processing approach in artificial intelligence, was enough to suggest to a large number of workers in the field that there was no future in perceptron-like computational devices for artificial intelligence and cognitive psychology. The problem is that although Minsky and Papert were perfectly correct in their analysis, the results apply only to these simple one-layer perceptrons and not to the larger class of perceptron-like models. In particular (as Minsky and Papert actually conceded), it can be shown that a multilayered perceptron system, including several layers of predicates between the retina and the decision stage, can compute functions such as parity, using reasonable numbers of units each computing a very local predicate. (See Chapters 5 and 8 for examples of multilayer networks that compute parity). Similarly, it is not difficult to develop networks capable of solving the connectedness or inside/outside problem. Hinton and Sejnowski have analyzed a version of such a network (see Chapter 7).

Essentially, then, although Minsky and Papert were exactly correct in their analysis of the *one-layer perceptron,* the theorems don't apply to systems which are even a little more complex. In particular, it doesn't apply to multilayer systems nor to systems that allow feedback loops.

Minsky and Papert argued that there would not be much value to multilayer perceptrons. First, they argued that these systems are sufficiently unrestricted as to be vacuous. They pointed out, for example, that a universal computer could be built out of linear threshold units.

Therefore, restricting consideration of machines made out of linear threshold units is no restriction at all on what can be computed.

We don't, of course, believe that the class of models sketched in Chapter 2 is a small or restrictive class. (Nor, for that matter, are the languages of symbol processing systems especially restrictive.) The real issue, we believe, is that different algorithms are appropriate to different architectural designs. We are investigating an architecture in which cooperative computation and parallelism is natural. Serial symbolic systems such as those favored by Minsky and Papert have a natural domain of algorithms that differs from those in PDP models. Not everything can be done in one step without feedback or layering (both of which suggest a kind of "seriality"). We have been led to consider models that have both of these features. The real point is that we seek algorithms that are *as parallel as possible.* We believe that such algorithms are going to be closer in form to the algorithms which could be employed by the hardware of the brain and that the kind of parallelism we employ allows the exploitation of multiple information sources and cooperative computation in a natural way.

A further argument advanced by Minsky and Papert against perceptron-like models with hidden units is that there was no indication how such multilayer networks were to be trained. One of the appealing features of the one-layer perceptron is the existence of a powerful learning procedure, the perceptron convergence procedure of Rosenblatt. In Minsky and Papert's day, there was no such powerful learning procedure for the more complex multilayer systems. This is no longer true. Chapters 5, 6, 7, and 8 all provide schemes for learning in systems with hidden units. Indeed, Chapter 8 provides a direct generalization of the perceptron learning procedure which can be applied to arbitrary networks with multiple layers and feedback among layers. This procedure can, in principle, learn arbitrary functions including, of course, parity and connectedness.

The problem of stimulus equivalence. A second problem with early PDP models—and one that is not necessarily completely overcome by multilayer systems—is the problem of invariance or *stimulus equivalence.* An *A* is an *A* is an *A*, no matter where on the retina it appears or how large it is or how it is oriented; and people can, in general, recognize patterns rather well despite various transformations. It has always seemed elegant and natural to imagine that an *A*, no matter where it is presented, is normalized and then processed for recognition using stored knowledge of the appearance of the letter (Marr, 1982; Neisser, 1967).

In conventional computer programs this seems to be a rather straightforward matter requiring, first, normalization of the input, and,

second, analysis of the normalized input. But in early PDP models it was never clear just how normalization could be made to work. Indeed, one of the main criticisms of perceptrons—one that is often leveled at more recent PDP models, too—is that they appear to provide no mechanism of attention, no way of focusing the machine on the analysis of a part of a larger whole and then switching to another part or back to the consideration of the whole.

While it is certainly true that certain PDP models lack explicit attentional mechanisms, it is far from true that PDP mechanisms are in principle incapable of exhibiting attentional phenomena. Likewise, while it is true that certain PDP models do not come to grips with the stimulus equivalence problem, it far from true that they are incapable of doing this in principle. To prove these points, we will describe a method for solving the stimulus equivalence problem that was described by Hinton (1981b). The idea is sketched in Figure 2. Essentially, it involves two sets of feature detectors. One (at the bottom of the figure) consists of *retinocentric* feature detectors and the other (above the retinocentric units) consists of canonical feature detectors. Higher order units that recognize canonical patterns (in this example, letters) sit above the canonical feature detectors and can have mutually excitatory connections to these feature detectors, just as in the interactive activation model of word recognition. What Hinton described was a method for mapping retinocentric feature patterns into canonical patterns. In general, for patterns in three-space, there are six degrees of freedom, but for present purposes we will consider only figures that are rotated around a fixed point in the plane. Here normalization simply amounts to a one-dimensional rotational transformation.

A fixed mapping from retinocentric units to canonical units would involve connecting each retinocentric feature detector to the corresponding canonical feature detector. Thus, to correct for a 90° clockwise rotation in the plane, we would want each retinal unit to project to the canonical unit corresponding to it at an offset of 90°.

How to implement variable mappings? Hinton proposed the use of a set of mapping units which act to switch on what amount to *dynamically programmable connections* from the retinocentric units to the canonical units. In the figure, three different mapping units are shown on the right: one that produces no rotation at all, one that produces a 90° clockwise rotation, and one that produces a 90° counterclockwise rotation. When one of these mapping units is active, it provides one of two inputs to a subset of the programmable connections. Thus, when the 90° clockwise mapping unit is active, it provides one of two inputs to the connection from each retinocentric unit to the central unit that corresponds to it under the 90° clockwise rotation. These connections are multiplicative—they pass the product of their two inputs on to the

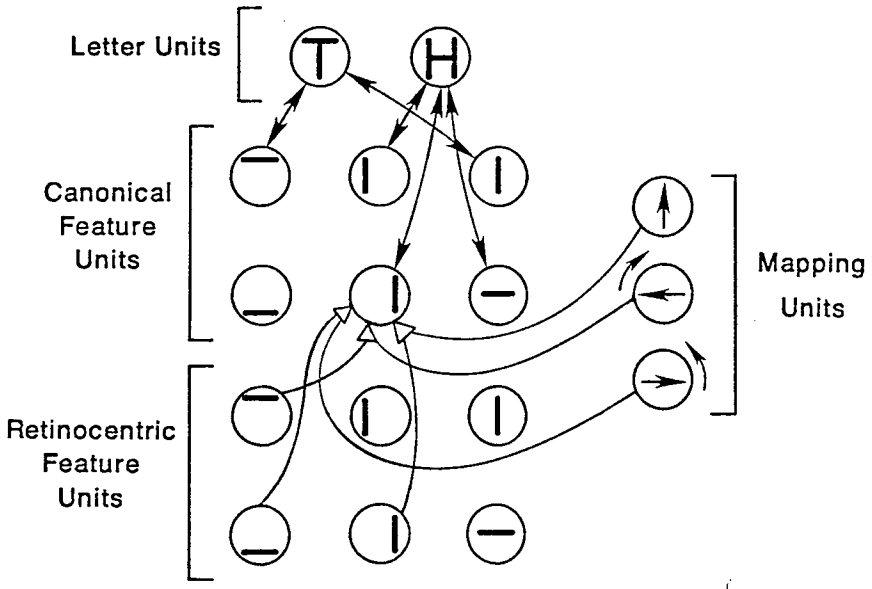

FIGURE 2. Hinton's (1981b) scheme for mapping patterns in one coordinate system into patterns in another coordinate system. At the top are two letter-detector units, with mutual excitatory connections to the six canonical feature units (the position and orientation of the line segment each of these detectors represents is indicated by the line segment in the "body" of each unit). At the bottom are six retinocentric feature units, and at the right are units corresponding to each of three different mappings from retinocentric to canonical features. (The arrows on the units indicate which direction in the retinocentric frame corresponds to upright in the canonical frame, and the arrow outside the unit indicates the nature of the transformation imposed on the retinocentric pattern). Each canonical unit receives three pairs of inputs, with each pair arriving at a multiplicative connection. These inputs are illustrated for one canonical unit only.

receiving unit. In this case, if a particular retinocentric feature is on and the 90° clockwise mapping unit is on, then the canonical feature corresponding to the active retinal feature will receive an excitatory input. If just one of the two inputs to the connection is on, no activation will flow to the central unit. In this way, when a mapping unit is active, it effectively programs the multiplicative connections needed to implement the corresponding mapping by activating one of the two inputs to each of the programmable connections.

Using this mechanism, it is possible to map from retinal to central coordinates if the mapping is known in advance. Object recognition can now proceed as follows: A mapping is chosen (perhaps on the basis of processing the preceding stimulus), and this is used to map a retinal input onto the canonical units. In a system involving variable

translational mappings, in addition to the rotational mappings shown here, it would be possible to focus the attention of the system successively on each of several different patterns merely by changing the mapping. Thus it would not be difficult to implement a complete system for sequential processing of a series of patterns using Hinton's scheme (a number of papers have proposed mechanisms for performing a set of operations in sequence, including Grossberg, 1978, and Rumelhart & Norman, 1982; the latter paper is discussed in Chapter 1).

So far, we have described what amounts to a PDP implementation of a conventional pattern recognition system. First, map the pattern into the canonical frame of reference, then recognize it. Such is the procedure advocated, for example, by Neisser (1967) and Marr (1982). The demonstration shows that PDP mechanisms are in fact capable of normalization and of focusing attention successively on one pattern after another.

But the demonstration may also seem to give away too much. For it seems to suggest that the PDP network is simply a method for implementing standard sequential algorithms of pattern recognition. We seem to be left with the question, what has the PDP implementation added to our understanding of the problem?

It turns out that it has added something very important. It allows us to begin to see how we could solve the problem of recognizing an input pattern even in the case where we do not know in advance either what the pattern is or which mapping is correct. In a conventional sequential algorithm, we might proceed by serial search, trying a sequence of mappings and looking to see which mapping resulted in the best recognition performance. With Hinton's mapping units, however, we can actually perform this search in parallel. To see how this parallel search would work, it is first necessary to see how another set of multiplicative connections can be used to choose the correct mapping for a pattern given both the retinal input and the correct central pattern of activation.

In this situation, this simultaneous activation of a central feature and a retinal feature constitutes evidence that the mapping that connects them is the correct mapping. We can use this fact to choose the mapping by allowing central and retinal units that correspond under a particular mapping to project to a common multiplicative connection on the appropriate mapping unit. Spurious conjunctions will of course occur, but the correct mapping units will generally receive more conjunctions of canonical and retinal features than any other (unless there is an ambiguity due to a symmetry in the figure). If the mapping units compete so that the one receiving the most excitation is allowed to win, the network can settle on the correct mapping.

We are now ready to see how it may be possible to simultaneously settle on a mapping and a central representation using both sets of

multiplicative connections. We simply need to arrange things so that when the retinal input is shown, each possible mapping we wish to consider is partially active. Each retinal feature then provides partial activation of the canonical feature corresponding to it under each of the mappings. The correct mapping allows the correct canonical pattern to be partially activated, albeit partially obscured by noise generated by the other partially activated mappings. Interactive activation between this central pattern and higher level detectors for the pattern then reinforces the elements of the pattern relative to the noise. This process by itself can be sufficient for correct recognition. Further cleanup of the central pattern can be achieved, though, by allowing the pattern emerging on the central units to work together with the input pattern to support the correct mapping over the other partially active mappings via the multiplicative connections onto the mapping units. This then results in further suppression of the noise. As this process continues, it eventually locks in the correct interpretation of the pattern, the correct canonical feature representation, *and* the correct mapping, all from the retinal input alone. Prior activation of the correct mapping facilitates the process of settling in, as do prior cues to the identity of the figure (see Rock, 1973, and Palmer, 1980, for evidence that these clues do facilitate performance), but are not, in general, essential unless the input is in fact ambiguous without them.

Hinton's mapping scheme allows us to make two points. First, that parallel distributed processing is in fact compatible with normalization and focusing of attention; and second, that a PDP implementation of a normalization mechanism can actually produce a computational advantage by allowing what would otherwise be a painful, slow, serial search to be carried out in a single settling of a parallel network. In general, Hinton's mapping system illustrates that PDP mechanisms are not restricted to fixed computations but are quite clearly capable of modulation and control by signals arising from other parts of an integrated processing system; and that they can, when necessary, be used to implement a serial process, in which each of several patterns is considered, one at a time.

The introduction of multiplicative or contingent connections (Feldman & Ballard, 1982) is a way of greatly increasing the power of PDP networks of fixed numbers of units (Marr, 1982; Poggio & Torre, 1978; see Chapter 10). It means, essentially, that each unit can perform computations as complex as those that could be performed by an entire one-layer perceptron, including both the predicates and the decision unit. However, it must also be noted that multiplicative connections are not strictly necessary to perform the required conjunctive computational operations. Nonlinear, quasi-multiplicative interactions can be implemented in a variety of ways. In the worst case, whole units could

be dedicated to each multiplicative operation (as in the predicate layer of the perceptron).[1]

While Hinton's mapping mechanism indicates how attention might be implemented in PDP systems and imports some of the power of parallel distributed processing into the problem of simultaneously solving the mapping problem and the recognition problem, it does leave something to be desired. This is the fact that it allows only a single input pattern to be processed at one time since each pattern must be mapped separately onto the canonical feature units. Serial attention is sometimes required, but when we must resort to it, we lose the possibility of exploiting simultaneous, mutual constraints among several patterns. What has been processed before can still influence processing, but the ensemble of to-be-processed patterns cannot exert simultaneous, mutual influence on each other.

There is no doubt that sequentiality is forced upon us in some tasks—precisely those tasks in which the thought processes are extended over several seconds or minutes in time—and in such cases PDP mechanisms should be taken to provide potential accounts of the internal structure of a process evolving in time during the temporally extended structure of the thought process (see Chapter 14). But, in keeping with our general goals, we have sought to discover ways to maximally exploit simultaneous mutual constraints—that is, to maximize parallelism.

One mechanism which appears to make some progress in this direction is the connection information distribution mechanism described in Chapter 16. That mechanism uses multiplicative connections like those used in Hinton's model to send connection information out from a central knowledge store so that it can be used in local processing networks, each allocated to the contents of a different display location. The mechanism permits multiple copies of the same knowledge to be used at the same time, thereby effectively allowing tokens or local copies of patterns to be constructed from centrally stored knowledge of types in a parallel distributed processing system. These tokens then can interact with each other, allowing several patterns, all processed using the same centrally stored information, to exert simultaneous, mutual constraints on each other. Since these ideas, and their relation to attention, are discussed at length in Chapter 16, we will not elaborate on them further here.

---

[1] The linear threshold unit provides a quasi-multiplicative combination rule, and Sejnowski (1981) has described in detail how close approximation of the quantitative properties of multiplication of signals can be achieved by units with properties very much like those observed in real neurons.

Recursion. There are many other specific points that have been raised with respect to existing PDP models. Perhaps the most common one has to do with recursion. The ability to perform recursive function calls is a major feature of certain computational frameworks, such as augmented transition network (ATN) parsers (Woods, 1973; Woods & Kaplan, 1971), and is a property of such frameworks that gives them the capability of processing recursively defined structures such as sentences, in which embedding may produce dependencies between elements of a surface string that are indefinitely far removed from each other (Chomsky, 1957). It has often been suggested that PDP mechanisms lack the capacity to perform recursive computations and so are simply incapable of providing mechanisms for processing sentences and other recursively defined structures.

As before, these suggestions are simply wrong. As we have already seen, one can make an arbitrary computational machine out of linear threshold units, including, for example, a machine that can carry out all the operations necessary for implementing a Turing machine; the one limitation is that real biological systems cannot be Turing machines because they have finite hardware. In Chapter 14, however, we point out that with external memory aids (such as paper and pencil and a notational system) such limitations can be overcome as well.

We have not dwelt on PDP implementations of Turing machines and recursive processing engines because we do not agree with those who would argue that such capabilities are of the essence of human computation. As anyone who has ever attempted to process sentences like "The man the boy the girl hit kissed moved" can attest, our ability to process even moderate degrees of center-embedded structure is grossly impaired relative to that of an ATN parser. And yet, the human ability to use semantic and pragmatic contextual information to facilitate comprehension far exceeds that of any existing sentence processing machine we know of.

What is needed, then, is not a mechanism for flawless and effortless processing of center-embedded constructions. Compilers of computer languages generally provide such facilities, and they are powerful tools, but they have not demonstrated themselves sufficient for processing natural language. What is needed instead is a parser built from the kind of mechanism which facilitates the simultaneous consideration of large numbers of mutual and interdependent constraints. The challenge is to show how those processes that others have chosen to explain in terms of recursive mechanisms can be better explained by the kinds of processes natural for PDP networks.

This challenge is one that has not yet been fully met. However, some initial steps toward a PDP model of language processing are described in Chapter 19. The model whose implementation is

described in that chapter illustrates how a variety of different constraints may be combined by PDP models to aid in the assignment of underlying roles to the constituents of sentences. The chapter also provides a discussion of three different ways in which the model could be extended to process embedded clauses in a way that is roughly consistent with human capabilities and limitations in this regard.

We do not claim to have solved these problems. Our existing models have limitations and much remains to be done. Our explorations have just begun. The question is not, is the job done—no computational framework can claim much on this score. The question instead is, can more progress be made through further exploration of the PDP perspective on the microstructure of cognition? The discovery of multilayer learning rules, the use of multiplicative connections to implement transformations of input patterns, the distribution of connection information, and the host of other developments described throughout this book, indicate to us that the answer to the question is "yes."

PDP Models Are Not Cognitive

We have observed that the cooperative character of parallel distributed processing often allows us to account for behavior which has previously been attributed to the application of specific rules of grammar or rules of thought. This has sometimes led us to argue that lawful behavior is not necessarily *rule-driven* behavior. Here, we must distinguish between *rules* and *regularities.* The bouncing ball and the orbiting planet exhibit regularities in their behavior, but neither is applying rules. We have demonstrated the power of this approach in our earlier work on word perception (McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1982) and on the learning of English morphology (Chapter 18). In these cases we have been able to show how the apparent application of rules could readily *emerge* from interactions among simple processing units rather than from application of any higher level rules.

Some have viewed our argument against explicit rules as an argument against the cognitive approach to psychology. We do not agree. We believe that we are studying the *mechanisms* of cognition. The application of a rule (e.g., the firing of a production) is neither more nor less cognitive than the activation of our units. The real character of cognitive science is the attempt to explain mental phenomena through an understanding of the mechanisms which underlie those phenomena.

A related claim that some people have made is that our models appear to share much in common with behaviorist accounts of behavior. While they do involve simple mechanisms of learning, there is a crucial difference between our models and the radical behaviorism of Skinner and his followers. In our models, we are explicitly concerned with the problem of internal representation and mental processing, whereas the radical behaviorist explicitly denies the scientific utility and even the validity of the consideration of these constructs. The training of hidden units is, as is argued in Chapters 5 to 8, the construction of internal representations. The models described throughout the book all concern internal mechanisms for activating and acquiring the ability to activate appropriate internal representations. In this sense, our models must be seen as completely antithetical to the radical behaviorist program and strongly committed to the study of representation and process.

PDP Models Are the Wrong Level of Analysis

It is sometimes said that although PDP models are perfectly correct, they are at the wrong level of analysis and therefore not relevant to psychological data. [2] For example, Broadbent (1985) has argued that psychological evidence is irrelevant to our argument about distributed memory because the distribution assumption is only meaningful at what Marr (1982) has called the *implementational* (physiological) level and that the proper psychological level of description is the *computational* level.

The issues of levels of analysis and of theorizing is difficult and requires a good deal of careful thought. It is, we believe, largely an issue of scientific judgement as to what features of a lower level of analysis are relevant to a higher one. We are quite sure that it is not a matter for prescription. We begin our response to this objection with a review of Marr's analysis and his three levels of description. We then suggest that indeed our models are stated at the same level (in Marr's sense) as most traditional models from cognitive science. We then describe other senses of levels, including one in which higher level accounts can be said to be convenient approximations to lower level accounts. This sense comes closest to capturing our view of the

[2] The following discussion is based on a paper (Rumelhart & McClelland, 1985) written in response to a critique by Donald Broadbent (1985) on our work on distributed memory (cf. Chapter 17 and McClelland & Rumelhart, 1985).

relation between our PDP models and other traditional information processing models.

Marr's Notion of Levels

David Marr (1982) has provided an influential analysis of the issue of levels in cognitive science. Although we are not sure that we agree entirely with Marr's analysis, it is thoughtful and can serve as a starting point. Whereas Broadbent acknowledges only two levels of theory, the computational and the implementational, Marr actually proposes three, the *computational*, the *algorithmic*, and the *implementational* levels. Table 1 gives a description of Marr's three levels. We believe that PDP models are generally stated at the algorithmic level and are primarily aimed at specifying the representation of information and the processes or procedures involved in cognition. Furthermore, we agree with Marr's assertions that "each of these levels of description will have their place" and that they are "logically and causally related." Thus, no particular level of description is independent of the others. There is an implicit computational theory in PDP models as well as an appeal to certain implementational (physiological) considerations. We believe this to be appropriate. It is clear that different algorithms are more naturally implemented on different types of hardware and, therefore, information about the implementation can inform our hypotheses at the algorithmic level.

TABLE 1

THE THREE LEVELS AT WHICH ANY MACHINE CARRYING OUT INFORMATION PROCESSING TASKS MUST BE UNDERSTOOD

Computational Theory	Representation and Algorithm	Hardware Implementation
What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?	How can this computational theory be implemented? In particular, what is the representation for the input and output, and what is the algorithm for the transformation?	How can the representation and algorithm be realized physically?

Note. From *Vision* by D. Marr, 1982, San Francisco: W. H. Freeman. Copyright 1982 by W. H. Freeman. Reprinted by permission.

Computational models, according to Marr, are focused on a formal analysis of the problem the system is solving—not the methods by which it is solved. Thus, in linguistics, Marr suggests that Chomsky's (1965) view of a *competence* model for syntax maps most closely onto a *computational* level theory, whereas a psycholinguistic theory is more of a *performance* theory concerned with how grammatical structure might actually be computed. Such a theory is concerned with the algorithmic level of description. It is the algorithmic level at which we are concerned with such issues as efficiency, degradation of performance under noise or other adverse conditions, whether a particular problem is easy or difficult, which problems are solved quickly and which take a long time to solve, how information is represented, etc. These are all questions to which psychological inquiry is directed and to which psychological data is relevant. Indeed, it would appear that this is the level to which psychological data speaks most strongly. At the computational level, it does not matter whether the theory is stated as a program for a Turing machine, as a set of axioms, or as a set of rewrite rules. It does not matter how long the computation takes or how performance of the computation is affected by "performance" factors such as memory load, problem complexity, etc. It doesn't matter how the information is represented, as long as the representation is rich enough, in principle, to support computation of the required function. The question is simply *what function* is being computed, not *how* is it being computed.

Marr recommends that a good strategy in the development of theory is to begin with a careful analysis of the goal of a particular computation and a formal analysis of the problem that the system is trying to solve. He believes that this top-down approach will suggest plausible algorithms more effectively than a more bottom-up approach. Thus, the computational level is given some priority. However, Marr certainly does not propose that a theory at the computational level of description is an adequate psychological theory.

As psychologists, we are committed to an elucidation of the algorithmic level. We have no quarrel with Marr's top-down approach as a strategy leading to the discovery of cognitive algorithms, though we have proceeded in a different way. We emphasize the view that the various levels of description are interrelated. Clearly, the algorithms must, at least roughly, compute the function specified at the computational level. Equally clearly, the algorithms must be computable in amounts of time commensurate with human performance, using the kind and amount of hardware that humans may reasonably be assumed to possess. For example, any algorithm that would require more specific events to be stored separately than there are synapses in the brain should be given a lower plausibility rating than those that require much less storage. Similarly, in the time domain, those algorithms that

would require more than one serial step every millisecond or so would seem poor candidates for implementation in the brain (Feldman & Ballard, 1982).

In short, the claim that our models address a fundamentally different level of description than other psychological models is based on a failure to acknowledge the primary level of description to which much psychological theorizing is directed. At this level, our models should be considered as *competitors* of other models as a means of explaining psychological data.

Other notions of levels. Yet we do believe that in some sense PDP models are at a different level than other cognitive models such as prototype theories or schema theory. The reason is that there is more between the computational and the implementational levels than is dreamt of, even in Marr's scheme. Many of our colleagues have challenged our approach with a rather different conception of levels borrowed from the notion of levels of programming languages. It might be argued that a model such as, say, schema theory or the ACT* model of John R. Anderson (1983) is a statement in a "higher level" language analogous, let us say, to the Pascal or LISP programming languages and that our distributed model is a statement in a "lower level" theory that is, let us say, analogous to the assembly code into which higher level programs can be compiled. Both Pascal and assembler, of course, are considerably above the hardware level, though the latter may in some sense be closer to the hardware and more machine dependent than the former.

From this point of view one might ask why we are mucking around trying to specify our algorithms at the level of assembly code when we could state them more succinctly in a high-level language. We believe that most people who raise the levels issue with regard to our models have a relationship something like this in mind. People who adopt this notion have no objection to our models. They only believe that psychological models are more simply and easily stated in an equivalent higher level language—so why bother?

We believe that the programming language analogy is very misleading, unless it is analyzed more carefully. The relationship between a Pascal program and its assembly code counterpart is very special indeed. It is necessary for the Pascal and assembly language to map *exactly* onto one another only when the program was *written* in Pascal and the assembly code was compiled from the Pascal version. Had the original "programming" taken place in assembler, there is no guarantee that such a relationship would exist. Indeed, Pascal code will, in general, compile into only a small fraction of the possible assembly code programs that could be written. Since there is every reason to suppose

that most of the programming that might be taking place in the brain is taking place at a "lower level" rather than a "higher level," it seems unlikely that some particular higher level description will be identical to some particular lower level description. We may be able to capture the actual code approximately in a higher level language—and it may often be useful to do so—but this does not mean that the higher level language is an adequate characterization.

There is still another notion of levels which illustrates our view. This is the notion of levels implicit in the distinction between Newtonian mechanics on the one hand and quantum theory on the other.[3] It might be argued that conventional symbol processing models are macroscopic accounts, analogous to Newtonian mechanics, whereas our models offer more microscopic accounts, analogous to quantum theory. Note, that over much of their range, these two theories make precisely the same predictions about behavior of objects in the world. Moreover, the Newtonian theory is often much simpler to compute with since it involves discussions of entire objects and ignores much of their internal structure. However, in some situations Newtonian theory breaks down. In these situations we must rely on the microstructural account of quantum theory. Through a thorough understanding of the relationship between the Newtonian mechanics and quantum theory we can understand that the macroscopic level of description may be *only an approximation* to the more microscopic theory. Moreover, in physics, we understand just when the macrotheory will fail and the microtheory must be invoked. We understand the macrotheory as a useful formal tool by virtue of its relationship to the microtheory. In this sense the objects of the macrotheory can be viewed as *emerging* from interactions of the particles described at the microlevel.

The basic perspective of this book is that many of the constructs of macrolevel descriptions such as schemata, prototypes, rules, productions, etc. can be viewed as emerging out of interactions of the microstructure of distributed models. These points are most explicitly considered in Chapters 6, 14, 17, and 18. We view macrotheories as approximations to the underlying microstructure which the distributed model presented in our paper attempts to capture. As approximations they are often useful, but in some situations it will turn out that an examination of the microstructure may bring much deeper insight. Note for example, that in a conventional model of language acquisition, one has to make very delicate decisions about the exact circumstances under which a new rule will be added to the rule system. In our PDP models no such decision need be made. Since the analog to a rule is

[3] This analogy was suggested to us by Paul Smolensky.

not necessarily discrete but simply something that may emerge from interactions among an ensemble of processing units, there is no problem with having the functional equivalent of a "partial" rule. The same observation applies to schemata (Chapter 14), prototypes and logogens (Chapter 18), and other cognitive constructs too numerous to mention. Thus, although we imagine that rule-based models of language acquisition—the logogen model, schema theory, prototype theory, and other macrolevel theories—may all be more or less valid approximate macrostructural descriptions, we believe that the actual algorithms involved cannot be represented precisely in any of those macrotheories.

It may also be, however, that some phenomena are too complex to be easily represented as PDP models. If these phenomena took place at a time frame over which a macrostructural model was an adequate approximation, there is no reason that the macrostructural model ought not be applied. Thus, we believe that the concepts of symbols and symbol processing can be very useful. Such models may sometimes offer the simplest accounts. It is, however, important to keep in mind that these models are approximations and should not be pushed too far. We suspect that when they are, some account similar to our PDP account will again be required. Indeed, a large part of our own motivation for exploring the PDP approach came from the failure of schema theory to provide an adequate account of knowledge application even to the task of understanding very simple stories.

Lest it may seem that we have given too much away, however, it should be noted that as we develop clearer understandings of the microlevel models, we may wish to formulate rather different macrolevel models. As pointed out in Chapter 3, PDP mechanisms provide a powerful alternative set of macrolevel primitives.[4]

Imagine a computational system that has as a primitive, "Relax into a state that represents an optimal global interpretation of the current input." This would be, of course, an extremely powerful place to begin building up a theory of higher level computations. Related primitives would be such things as "Retrieve the representation in memory best matching the current input, blending into it plausible reconstructions of details missing from the original memory trace," and "Construct a dynamic configuration of knowledge structures that captures the present situation, with variables instantiated properly." These sorts of primitives would be unthinkable in most conventional approaches to higher level cognition, but they are the kinds of emergent properties that PDP mechanisms give us, and it seems very likely that the availability of

[4] We thank Walter Schneider for stressing in his comments on an earlier draft of this chapter the importance of the differences between the computational primitives offered by PDP and those offered by other formalisms for modeling cognitive processes.

such primitives will change the shape of higher level theory considerably.

PDP mechanisms may also place some constraints on what we might realistically ask for in the way of computational primitives because of the costs of implementing certain kinds of computations in parallel hardware in a single relaxation search. The parallel matching of variablized productions is one case in point. Theories such as ACT* (J. R. Anderson, 1983) assume that this can be done without worrying about the implementation and, therefore, provide no principled accounts of the kinds of crosstalk exhibited in human behavior when processing multiple patterns simultaneously. However, it appears to be a quite general property of PDP mechanisms that they will exhibit crosstalk when processing multiple patterns in parallel (Hinton & Lang, 1985; Mozer, 1984; see Chapters 12 and 16).

High-level languages often preserve some of the character of the lower level mechanisms that implement them, and the resource and time requirements of algorithms drastically depends on the nature of the underlying hardware. Higher level languages that preserve the character of PDP mechanisms and exploit the algorithms that are effective descriptions of parallel networks are not here yet, but we expect such things to be coming along in the future. This will be a welcome development, in our view, since certain aspects of cognitive theory have been too strongly influenced by the discrete, sequential algorithms available for expression in most current high-level languages.

As we look closely, both at the hardware in which cognitive algorithms are implemented and at the fine structure of the behavior that these algorithms are designed to capture, we begin to see why it may be appropriate to formulate models which come closer to describing the microstructure of cognition. The fact that our microstructural models can account for many of the facts about the representation of general and specific information, for example, as discussed in Chapter 18, makes us ask why we should view constructs like logogens, prototypes, and schemata as anything other than convenient approximate descriptions of the underlying structure of memory and thought.

Reductionism and Emergent Properties

A slightly different, though related, argument is that the PDP enterprise is an exercise in reductionism—an exercise in which all of psychology is reduced to neurophysiology and ultimately to physics. It is argued that coherent phenomena which emerge at any level (psychology or physics or sociology) require their own language of description

and explanation and that we are denying the essence of what is cognitive by reducing it to units and connections rather than adopting a more psychologically relevant language in our explanations.

We do not classify our enterprise as reductionist, but rather as interactional. We understand that new and useful concepts emerge at different levels of organization. We are simply trying to *understand* the essence of cognition as a property emerging from the *interactions* of connected units in networks.

We certainly believe in emergent phenomena in the sense of phenomena which could never be understood or predicted by a study of the lower level elements in isolation. These phenomena are functions of the particular kinds of groupings of the elementary units. In general, a new vocabulary is useful to talk about aggregate phenomena rather than the characteristics of isolated elements. This is the case in many fields. For example, we could not know about diamonds through the study of isolated atoms; we can't understand the nature of social systems through the study of isolated individuals; and we can't understand the behavior of networks of neurons from the study of isolated neurons. Features such as the hardness of the diamond is understandable through the interaction of the carbon atoms and the way they line up. The whole is different than the *sum* of the parts. There are nonlinear interactions among the parts. This does not, however, suggest that the nature of the lower level elements is irrelevant to the higher level of organization—on the contrary, the higher level is, we believe, to be understood primarily through the study of the interactions among lower level units. The ways in which units interact is not predictable from the lower level elements as isolated entities. It is, however, predictable *if* part of our study involves the interactions among these lower level units. We *can* understand why diamonds are hard, not as an isolated fact, but because we understand how the atoms of carbon can line up to form a perfect lattice. This is a feature of the aggregate, not of the individual atom, but the features of the atom are necessary for understanding the aggregate behavior. Until we understand that, we are left with the unsatisfactory statement that diamonds are hard, period. A useful fact, but not an explanation. Similarly, at the social level, social organizations cannot be understood without understanding the individuals which make up the organization. Knowing about the individuals tells us little about the structure of the organization, but we can't *understand* the structure of the higher level organizations without knowing a good deal about individuals and how they function. This is the sense of emergence we are comfortable with. We believe that it is entirely consistent with the PDP view of cognition.

There is a second, more practical reason for rejecting radical reductionism as a research strategy. This has nothing to do with emergence;