TWO NOTIONS OF IMPLICIT RULES

Martin Davies
Corpus Christi College, University of Oxford

In popular accounts of the differences between connectionism and the (‘Classical’) rules and representations paradigm, it is often said that connectionist networks perform cognitive tasks without knowledge of rules. Thus, for example (Rumelhart and McClelland, 1986, p. 218):

We would...suggest that parallel distributed processing models may provide a mechanism sufficient to capture lawful behaviour, without requiring the postulation of explicit but inaccessible rules.

Equally, when the stress is upon the way that networks learn, it is said that networks learn to perform cognitive tasks without learning rules (McClelland, Rumelhart and Hinton, 1986, p. 32):

[W]e do not assume that the goal of learning is the formulation of explicit rules. Rather, we assume it is the acquisition of connection strengths which allow a network of simple units to act as though it knew the rules.

Likewise (Norman, 1986, p. 536):

[A]lthough the system develops neither rules of classification nor generalizations, it acts as if it had these rules... . It is a system that exhibits intelligence and logic, yet that nowhere has explicit rules of intelligence or logic.

These three quotations all come from early expositions of the connectionist programme (indeed, from the PDP ‘Bible’), but such claims recur frequently enough. We find something similar, for example, in a more recent description of a connectionist model of the cognitive task of reading single words aloud (Seidenberg and McClelland, 1989).

In this case, the connectionist model stands in contrast to a ‘dual route’ model of reading aloud, a model in which one route—crucial for the pronunciation of irregular words like ‘pint’ and ‘have’—makes use of a lexicon, while the other route is able to achieve the correct pronunciation of regular words (‘mint’, ‘slave’) and of non-words (‘slint’, ‘mave’) by way of letter-sound, or grapheme-
phoneme, or more generally sub-word level orthographic-to-phonological *rules* (Coltheart, 1985). Mark Seidenberg says of the connectionist reading aloud model (1989, p. 40):

In contrast to the dual-route model, there are no rules specifying the regular spelling-sound correspondences of the language and there is no phonological lexicon in which the pronunciations of all words are listed.

But it also worth noting that Seidenberg says later (1989, pp. 66-7):

What the model shows is that certain simple, intuitive notions of what is meant by ‘rule’ fail to capture relevant generalizations about naming [reading aloud] behaviour. ... It is valid to ask whether a particular notion of ‘rule’ is adequate as a means of capturing generalizations of a particular sort. ... However, it is vacuous to ask whether behavior is rule governed if the notion of ‘rule’ is unconstrained. Much of the debate over ‘rules’ to this point...has little force because no explicit notion of rule is at stake.

Seidenberg says ‘explicit notion of rule’, not ‘notion of explicit rule’; and the difference is important. If we restrict attention to the idea of an explicit rule (‘certain simple, intuitive notions of what is meant by “rule”’) then, although even that idea needs to be made more precise (Section 1.2 below), we can agree already that there are no spelling-sound rules in the Seidenberg and McClelland reading aloud network. But what Seidenberg invites is an attempt to make explicit some other notion, or notions, of rule for which it would at least be sensible to ask whether the network knows or embodies spelling-sound rules.

Scanning the connectionist literature, we may come upon sentences that seem, at first glance, flatly to contradict the ‘Biblical’ orthodoxy about networks and rules. Consider, for example (Bates and Elman, 1993, p. 634): ‘Contrary to rumor, it is not the case that connectionist systems have no rules.’ What notion of rules is at stake here? The answer from Bates and Elman is that (ibid.):

the ‘rules’ in a connectionist net include the connections that hold among units, i.e. the links or ‘weights’ that embody all the potential mappings from input to output across the system as a whole. This means that rules (like representations) can exist by degree, and vary in strength.

So, on this account, each weighted connection—from an input unit to a hidden unit, or from a hidden unit to an output unit—embodies a rule; and a network with thirty, or fifty, or a hundred units embodies hundreds or thousands of rules.

The Seidenberg and McClelland network with 400 input, 200 hidden, and 460 output units would, by this reckoning, embody 172,000 rules. Whatever might be the content of these rules, we can be confident that they will not all be sub-word level orthographic-to-phonological conversion rules of the kind envisaged in the dual route model. There are simply too many of them. And, in
fact, we can see that none of these rules—embodied in the connection between, say, a single input unit and a single hidden unit—can be a letter-sound rule, or a grapheme-phoneme rule, or any other kind of sub-word level orthographic-to-phonological rule. In the Seidenberg and McClelland model, the system for coding the orthographic input proceeds by, first, converting a word, such as ‘cat’, into a set of Wickelgraphs (trios of letters and spaces)—in this case, the set \{#CA, CAT, AT#\}. Then, each of the 200 input units is correlated with 1,000 Wickelgraphs, in such a way that each Wickelgraph activates about 20 input units. So, in short, no input unit corresponds to a letter, or to a grapheme, or to any sub-word unit that might be the subject of a spelling-sound rule.

Bates and Elman go on to say (1993, p. 634-5):

[It is possible for several different networks to reach the same solution to a problem, each with a totally different set of weights. This fact runs directly counter to the tendency in traditional cognitive and linguistic research to seek ‘the rule’ or ‘the grammar’ that underlies a set of behavioral regularities.

It is certainly true that the Bates and Elman notion of rule returns only a trivial answer to the question whether two versions of the Seidenberg and McClelland reading aloud network—the results of training runs from different random initial weights, say—embody the same spelling-sound rules. The trivial answer is that neither network embodies any such rules, in the Bates and Elman sense. And if two networks were to embody spelling-sound rules by the Bates and Elman criterion, then they would embody different rules, simply in virtue of having different weight matrices. But Bates and Elman speak tantalizingly of the possibility that two networks might reach ‘the same solution to a problem’, even though the actual weights in the networks are quite different. What is this notion of ‘the same solution’?

Now, it may be that all that Bates and Elman have in mind here is that two networks can achieve the same input-output relation on a training set, even though the networks are internally very different. But, what would be more interesting would be the idea that, amongst networks all of which achieve the correct input-output relation to solve a problem, the networks can be classified into groups on the basis of having reached the same solution—having adopted the same way of achieving the input-output relation, despite having different weights. This would be an interesting idea because it would suggest a level of classification of networks that would discriminate more finely than mere input-output relations, but more coarsely than weight matrices—a level of classification that might conceivably be inhabited by a new notion of rule.

Let us pause to review the situation thus far. Connectionist networks typically do not contain explicit rules in the way that classical AI systems do. Nevertheless, there is an invitation (Seidenberg, 1989) to develop new notions of rule so that we can raise the question whether a connectionist network embodies spelling-sound rules, for example. We might begin with the idea that rules are
embodied in weights on individual connections (Bates and Elman, 1993). But at that fineness of grain, we shall not find spelling-sound rules in the Seidenberg and McClelland network—nor, generally, shall we uncover in typical connectionist networks the kinds of rules that figure in ‘traditional cognitive and linguistic research’. What would be interesting would be a notion of rule that allows that a rule might indeed be embodied in a single weighted connection, but which also permits one and the same rule to be embodied in different configurations of weights in different networks.

Gary Hatfield has remarked that (1991, p. 90):

Connectionist approaches to cognition afford a new opportunity for reflection on the notions of rule and representation as employed in cognitive science.

In this paper, I shall be taking up that opportunity, so far as rules are concerned. The remainder of the paper is in four main sections. In Section 1, I review two extreme notions of knowledge of rules. In Section 2, I introduce an intermediate notion which makes use of the idea of a causally systematic process. This is my first notion of implicit rule. In Section 3, I distinguish this from a second notion of implicit rule, which imposes fewer requirements upon the causal processes taking place inside a system. Finally, in Section 4, I use the distinction between the two notions of implicit rule to shed light upon a recent dispute about structure-sensitive processes, between Jerry Fodor, Zenon Pylyshyn, and Brian McLauglin on the one hand (Fodor 1987; Fodor and Pylyshyn, 1988; Fodor and McLauglin, 1990), and Paul Smolensky on the other (Smolensky, 1991a, 1991b).

1. Two extreme notions of knowledge of rules

The two notions of knowledge of rules from which I begin are, at one extreme—the strong end of the spectrum—the idea of a rule that is explicitly represented within a system and, at the other extreme—the weak end of the spectrum—the idea of a rule to which the input-output relation of a system conforms, whether by design or by happenstance. The orthodoxy about connectionism is that networks achieve conformity to rules without explicit representation of rules. Let us take some time over each of these extreme notions.

1.1 Knowledge in the weak sense: Conformity

The notion of conformity to a rule is simple; but still, there is one possible misunderstanding to be avoided. Someone might say, naturally enough, that a wooden block sliding along a smooth surface conforms to the rule ‘F = m.a’ (‘force equals mass times acceleration’), or equivalently ‘a = F/m’ (‘acceleration equals force divided by mass’). But this is not the notion of conformity to a rule
that we need to have at the weak end of our spectrum.

We are considering information processing systems, and a system that possesses knowledge of a rule—whatever exactly that turns out to be—is a system that has a resource enabling it to perform inference-like transitions between input and output states that are states of information, that have content or meaning. Knowledge of the rule ‘a = F/m’ would permit the transition from an input state that means (contains or encodes the information) that the force is n units to an output state that means that the acceleration is n/m units. If we are asking about the presence or absence of that knowledge, then the relevant input-output transitions are not from an input state that is a force to an output state that is an acceleration, but from an input state that represents a force to an output state that represents an acceleration.

At the weak end of our spectrum of notions of knowledge of rules, then, we have the idea of a system that performs such input-output transitions, and does so in such a way that the resulting input-output pairs are, in point of their meaning, in conformity with the rule.

In a similar way, we can consider a spelling-sound rule, perhaps that an initial ‘c’, coming before ‘a’, ‘o’, or ‘u’, is pronounced /K/. A system that is performing the reading aloud task is presented with a token of a written word—a letter string—say ‘cat’, and produces a pronunciation—a phoneme string—say /KAT/. In some theoretical contexts, we might consider the written word token to be the input and the utterance to be the output. For our purposes, though, we shall take as the input state a state of the system that represents the orthographic item ‘cat’, and as the output state a state of the system that represents the pronunciation /KAT/. The transitions that interest us are then transitions from such input states to such output states. To the extent that the system yields input-output pairs that conform to the rule in point of their meaning—a representation of the letter string ‘cod’ gives rise to a representation of the phoneme string /KOD/, and so on—we shall say that the system has knowledge of the rule in the weak sense.

In order to raise the question whether a system possesses knowledge of a rule, even in this weak sense, we already have to presume upon the idea that the input and output states of the system have meaning—that they are representational states. In the Seidenberg and McClelland network, the input state that represents the written word ‘cat’ is a pattern of activation over about 60 of the input units. Similarly, the output state that represents the pronunciation /KAT/ is a pattern of activation over some of the output units. Hatfield suggests (1991, p. 90) that we need to reflect upon the notion of representation as well as the notion of rule; and that is right. We need some philosophical account of what it is, in virtue of which a pattern of activation counts as having meaning, or representing something. But for the purposes of this paper, and our reflections upon the notion of a rule, I shall take for granted the idea of input and output states that have representational properties.
1.2 *Knowledge in the strong sense: Explicit representation*

The notion of explicit representation of a rule is not entirely simple, since different theorists have chosen to label rather different ideas with the term ‘explicit’ (versus ‘implicit’ or ‘tacit’).

Michael Dummett says (1991, p. 96):

Someone has explicit knowledge of something if a statement of it can be elicited from him by suitable enquiry or prompting...

And (1991, p. 97):

A body of knowledge, however explicit, is obviously not continuously before our consciousness, being a store of items available, save when our memory betrays us, for use when needed. How the storage is effected is of no concern to philosophy: what matters to it is how each item is presented when summoned for use.

So, in Dummett’s usage, explicitness is a matter of the subject being able to present information in linguistic form, and is not a matter of how the information is stored in between presentations. Explicit knowledge is *ipso facto* accessible knowledge—‘save when our memory betrays us’.

The orthodoxy about connectionism says that networks lack explicit representations of rules, and contrasts the connectionist programme with ‘the explicit inaccessible rule view’ (Rumelhart and McClelland, 1986, p. 217). Since in Dummett’s usage, ‘explicit’ and ‘inaccessible’ are contradictory, we must look elsewhere for help with the notion of explicit representation that is under discussion here.

In a rather similar way, the distinction in experimental psychology between explicit and implicit memory tests does not help us. In an implicit memory test (Schacter, 1989, p. 695):

memory for a recent experience is inferred from facilitations of performance, generally known as repetition or direct priming effects, that need not and frequently do not involve any conscious recollection of the prior experience.

In contrast, explicit memory tests (ibid.):

make explicit reference to and demand conscious recollection of a specific previous experience.

Given that we are considering human information processing much of which is unconscious, and that we also want our notions of knowledge of rules to be applicable to systems—such as small connectionist networks—for which the question of conscious recollection does not even arise, this second usage of
'explicit' is of no help to us either.

The notion of explicit representation of rules that underwrites the connectionist orthodoxy—that networks do not contain explicit representations of rules—has at least two components. One component is the idea of a syntactically structured representation that encodes the rule. The second component is the idea of a state of knowledge whose presence in a system by itself falls very far short of explaining the input-output transitions for which the knowledge is logically adequate.

Thus, imagine that a system contains a representation of the rule ‘\(a = F/m\)’ in a linguistic format: the rule is written down on a sheet of paper, stored in one of many pigeon holes. And suppose that the system goes into an input state that encodes the information that \(F = 10\) units and \(m = 5\) units. The proposition that \(a = 2\) units follows logically from the propositions that \(a = F/m\), that \(F = 10\) units, and that \(m = 5\) units. But, in the system, a great deal needs to happen if an output state encoding the information that \(a = 2\) units is to be produced. The relevant sheet of paper needs to be located in its pigeon hole (search), and it needs to be brought together with the input representations in the workspace (access). There, some inferences need to be carried out, and those inferences require the arithmetical information that \(10 \div 5 = 2\). Perhaps that information is somehow ‘built into’ the mechanism that subserves the inference process; but on the other hand, it might itself be stored on a sheet of paper in a pigeon hole; or it might have to be inferred from some generalization that is stored in that way. However the details may go, it is clear that, in order to make use of the knowledge that is stored in it, the system must go through processes of search and access, as well as logical and arithmetical processes.

Both components of the idea of explicit representation are present in the example that I have described. There is a syntactically structured representation of the rule (it is written down on paper); and there is a considerable gap between having the knowledge (stored in a pigeon hole) and using it (to derive the result that \(a = 2\) units). But the presence of this kind of gap (the second component) does not depend essentially upon what is stored being a representation in a linguistic format (the first component).

To see this, just imagine a variant of the example where what is stored in the pigeon hole is not a sheet of paper with ‘\(a = F/m\)’ written on it, but instead a little machine. If the representations of the force and the mass are plugged into this machine, it produces a representation of the acceleration. But merely having representations of the force and the mass as input states of the overall system is not, by itself, enough to engage the little machine; first, it has to be located and moved into position. If there were a problem in finding the little machine in its pigeon hole, then the overall system would not produce an output state representing the acceleration. So, in this variant example, there is still a gap between having the knowledge (embodied in the stored machine) and using it to mediate input-output transitions.

In describing the variant example, I have helped myself to the notion of the
stored machine embodying knowledge of the rule ‘a = F/m’. That notion cries out for explanation, of course. But the point of highlighting the gap between having knowledge and using it is that it leads us to a contrasting feature of connectionist networks. For in networks there is no such gap (McClelland, Rumelhart and Hinton, 1986, p. 32; italics added):

The representation of the knowledge is set up in such a way that the knowledge necessarily influences the course of processing. Using knowledge in processing is no longer a matter of finding the relevant information in memory and bringing it to bear; it is part and parcel of the processing itself.

So, the connectionist orthodoxy that networks do not contain explicit representations of rules itself has two elements. First, networks do not contain stored representations of rules in a linguistic format. Second, and more generally, networks store their knowledge in a way that does not present any problem of search and access; rather, it is ‘part and parcel of the processing’.

The strong notion of knowledge of rules as explicit representation enables us to draw a distinction within the class of systems whose input-output transitions are in conformity with a rule (that is, which have knowledge of a rule in the weak sense). On one side of this distinction, we can place symbol manipulation devices which make use of a stored syntactically structured representation of the rule in question—rule-explicit symbol manipulation devices. On the other side of the distinction we find a heterogeneous collection of systems. There are symbol manipulation devices that do not make use of a syntactically structured representation of the rule. There are connectionist networks. And there are look-up tables that simply store all the input-output pairings independently.

In the case of models of the reading aloud task, we would have on one side of the distinction those models that make use of syntactically structured representations of spelling-sound rules. Within such a model, there would need to be additional machinery to locate the correct rules to apply given a particular input, and then to draw out the logical consequences of those rules for that input. The implementation of the non-lexical part of the dual route model by Coltheart, Curtis, Atkins and Haller (1993) is essentially of this type.

On the other side of the distinction, we would have models containing a host of little processing mechanisms corresponding to grapheme-phoneme conversion rules. (Once again, there would need to be extra machinery to bring the correct mechanisms into operation upon any given input.) We would have interactive activation models (McClelland and Rumelhart, 1981; Taft, 1991; also the implementation of the lexical route of the model envisaged by Coltheart et al., 1993), and also connectionist models, such as the Seidenberg and McClelland network and the earlier NETtalk of Sejnowski and Rosenberg (1987). And finally, we would have systems that operate by table look-up.
1.3 The next step

Many different information processing systems might achieve conformity to a rule, or to a set of rules, over some domain of examples. That is, many systems might have knowledge of the rule or rules in the weak sense. Within that class of systems, some might operate by having knowledge of the rule or rules in the strong sense: they might be rule explicit. What we are seeking is a new notion of knowledge of rules which would allow us sensibly to ask whether a connectionist network has knowledge of spelling-sound rules, for example. We are looking to make a distinction within the class of connectionist networks, and more generally within the class of systems that achieve conformity to the rules without being rule explicit.

Fodor and Pylyshyn remark (1988, p. 60):

[O]ne should not confuse the rule-implicit/rule-explicit distinction with the distinction between Classical and Connectionist architecture. ...

The one thing that Classical theorists do agree about is that it can't be that all behavioral regularities are determined by explicit rules; at least some of the causal determinants of compliant behavior must be implicit...

Classical machines can be rule implicit with respect to their programs,...

Unfortunately, Fodor and Pylyshyn are not absolutely explicit about what 'rule implicit' means, but they do talk about some of the functions of a computer being 'wired in'—we might say, being 'part and parcel of the processing'. So there is at least the suggestion here of a notion of rule implicit system that might include some classical symbol manipulation devices and also some connectionist networks.

Perhaps all that Fodor and Pylyshyn intend by 'rule implicit' is a machine that achieves input-output transitions in conformity with certain rules without explicitly representing those rules. But, however that may be, my plan is to introduce two notions of implicit rules that are intermediate between the strong and weak notions of knowledge of rules. They are intermediate in the sense that to say that a rule is implicit in a system will require more than mere conformity to the rule but less than explicit representation of the rule.

2. The first notion of implicit rules

The notion of implicit rules to be introduced in this section makes use of the idea of a causally systematic process. Although many refinements would be needed in a full treatment, the core of this idea is very simple. (For some of the complications, see Davies, 1987. For applications of the idea to connectionist networks, see Davies, 1989, 1990a, 1990b, 1991.)
2.1 Causally systematic processes

To begin with, we note that the weak notion of knowledge of rules—the notion of conformity—can be formulated in terms of a rule’s describing a pattern in a system’s input-output relation. In our earlier example, an information processing system performs transitions from input states that represent a force and a mass to output states that represent an acceleration. Where the input-output transitions are in conformity with the rule ‘\(a = F/m\)’, that rule describes a pattern in the input-output relation. To say that there is this pattern in the input-output relation is to say nothing, yet, about the way in which the input-output transitions that instantiate the pattern are performed—nothing about the structure of the processing that takes place inside the system.

The crucial diagnostic question for the first notion of implicit rules concerns the causal explanation of these input-output transitions. Do the transitions that instantiate the pattern have a common causal explanation? A roughly equivalent way of asking the same question is this. Is there, within the system, a component mechanism, or processor, or module that operates as a causal common factor to mediate all the input-output transitions that instantiate the pattern described by the rule? If so, then the rule is said to be implicit in the system (or the system is said to have implicit or tacit knowledge of the rule).

We can add two small points of clarification here. First, it might be better to say that, if the answer to our question is ‘Yes’, then the rule is at least implicit in the system, for the case where the rule ‘\(a = F/m\)’ is explicitly represented is intended to be one case in which the question is answered affirmatively. Second, it is important that the causal common factor really does mediate the transitions from input state through to output state. It is certainly not sufficient for implicit knowledge of a rule that there should merely be some tiny component that figures somewhere in all the transitions.

Consider, then, the system that we described earlier, in which the rule ‘\(a = F/m\)’ is written down on a sheet of paper and stored in a pigeon hole. When the system goes into the input state that encodes the information that \(F = 10\) units and \(m = 5\) units, the resulting output state encodes the information that \(a = 2\) units. When the input state means that \(F = 12\) units and \(m = 4\) units, the resulting output state says that \(a = 3\) units. These two input-output pairs instantiate the pattern described by the rule. Do the two transitions have a common causal explanation? The answer is ‘Yes’ because the core of the explanation of both transitions is afforded by the presence of the stored representation of the rule. In each case, the representation of the rule is located, brought together with the input state, and subjected to inferential procedures. In fact, all the input-output transitions that are in conformity with the rule have the same causal explanation.

The answer would be ‘No’ in the case of a system that operated by table look-up. In such a case, the explanations of the two input-output transitions would be utterly different. Given the first input state, a search procedure would
operate to locate a sheet of paper with ‘F = 10, m = 5 : ∴ ___’ written on it, and what follows the ‘∴’—namely, ‘a = 2’—would be used to generate the output state. The presence of that sheet of paper in its pigeon hole would be fundamental in the causal explanation of the first input-output transition, but would be explanatorily irrelevant to the performance of the second transition.

Thus, we have three notions of knowledge of rules: the strong notion, the weak notion, and the intermediate notion—our first notion of implicit rules. But the two examples that we have just described are not enough to show that the first notion of implicit rules is genuinely intermediate between the two extreme notions. For, on those two examples, the answer to our diagnostic question goes in step with the presence or absence of an explicit representation of the rule. What we need is an example where the answer to our diagnostic question is ‘Yes’ even though the rule is not explicitly represented.

2.2 Three configurations

In effect, we have already indicated an example of the type that we need, in our discussion of models of reading single words aloud. Along with a model that contains explicit representations of spelling-sound rules, and a model that operates by table look-up, we mentioned the possibility of a model containing processing mechanisms corresponding to the various grapheme-phoneme conversion rules. However, in order to be more explicit about an example, and also to indicate how the first notion of implicit rules allows us to make a distinction within the class of connectionist networks, let us consider a very simplified version of the reading aloud task.

The task domain consists of just twenty five items, each of which is a two-letter string. In each string, the first letter is one of the consonants, ‘b’, ‘d’, ‘f’, ‘h’, ‘k’, and the second letter is one of the vowels ‘a’, ‘e’, ‘i’, ‘o’, ‘u’. The pronunciation of these twenty five items is completely regular, and we indicate the phonemes corresponding to the ten letters by using capital letters. There are ten letter-sound rules that characterize this reading aloud task, and we shall consider three models that achieve conformity with those ten rules. One model, which we shall not describe in any more detail, is a rule-explicit system—as it might be, a much simplified version of the Coltheart et al., (1993) model.

The rule-explicit system, we may suppose, performs in conformity with the ten letter-sound rules, so it has knowledge of those rules in the weak sense. Since it is a rule-explicit system, it also has knowledge of the rules in the strong sense. And since the presence of an explicit representation is sufficient for an affirmative answer to our diagnostic question, the rules are implicit (or at least implicit) in the system according to our first notion of implicit rules. The other two models are both very simple connectionist networks.

The first network has ten input units, in two pools of five, and likewise ten output units. A two-letter string is represented by a pattern of activation involving two of the ten input units, one from each pool. The string ‘ba’ is represented
by the vector of activations \langle 1, 0, 0, 0, 0, 1, 0, 0, 0, 0 \rangle across the ten input units. Similarly, a two-phoneme string is represented by activation at two of the output units. There are no hidden units, and each input unit is connected (with a weight of 1) to exactly one output unit. Figure 1 shows the input state that represents the letter string ‘ba’ and the output state that represents the phoneme string /BA/ in this network—which we call the modestly modular configuration.

The second network has twenty five input units, corresponding to the items in the task domain, and likewise twenty five output units. Each two-letter string is thus represented by activation at just one input unit. Each input unit is connected to just one output unit, activation at which represents the corresponding phoneme string. Figure 2 shows the input state that represents the letter string ‘ba’, and the output state that represents the phoneme string /BA/ in this network—which we call the madly modular configuration.

Each network performs an input-output transition for each of the twenty five items in the task domain, and the twenty five input-output pairs conform to the ten letter-sound rules. So, although neither network has knowledge of those ten rules in the strong sense, each has knowledge of the rules in the weak sense. Each of those ten rules describes a pattern that is instantiated five times amongst the twenty five input-output pairs. Thus, for example, the letter-sound rule that says that any two-letter string ‘b_’ is pronounced /B_/ describes a pattern instantiated by the five pairs: \langle ba', /BA/ \rangle, \langle be', /BE/ \rangle, ..., \langle bu', /BU/ \rangle. So, we can ask our diagnostic question about those five input-output transitions. Do they, in the respect in which they conform to the letter-sound rule for ‘b’, have a common causal explanation?

In the case of the modestly modular configuration, the answer is ‘Yes’. The connection between the input unit for ‘b’ and the output unit for /B/ figures as a causal common factor in the five transitions. In the madly modular configuration, the answer is ‘No’. The connection that is causally responsible for the input-output transition for ‘ba’ is explanatorily quite irrelevant to the other four transitions.

The madly modular configuration is essentially a connectionist version of table look-up, and we already had an example to illustrate that where there is table look-up there are no implicit rules (except the trivial rules that have only one instance each). But it is the modestly modular configuration that provides the example that we needed: the answer to our diagnostic question is ‘Yes’ for each of the ten letter-sound rules, although none of those rules is explicitly represented in the network. That is enough to show that the first notion of implicit rules is genuinely intermediate between the two extreme notions of knowledge of rules. And, despite the fact that the two networks are exceedingly simple—particularly in that they lack hidden units—they do illustrate that the first notion of implicit rules can, in principle, make distinctions within the class of connectionist networks.
2.3 Adding hidden units

We can, in fact, move quite simply from the modestly modular network to a network with hidden units which has the same spelling-sound rules implicit in it. Figure 3 shows a network that uses the same input and output coding schemes as the modestly modular network, but also includes ten hidden units. The input activation pattern that represents the letter string ‘ba’, results in activation at just two hidden units, and then at the corresponding output units, yielding the output pattern that represents /BA/. There are various different ways in which this pattern of hidden unit activation might be achieved. In one variation, each of the hidden units has a level of activation that is simply equal to the total activation coming into it (a linear activation function). The connection strengths from each input unit to just one hidden unit, and from that hidden unit to the corresponding output unit are one, and all other connection strengths are zero. An alternative would be for the hidden units to be simple threshold units, turned on by net incoming activation of more than 0.5, say. In that case, the connection strengths that were zero in the first variation could be non-zero, so long as they remained small. And the connection strengths that were one could now be less than one, so long as they remained large enough.

Whatever the details, the two input-hidden connections and two hidden-output connections that are shown in bold explain the input-output transition for ‘ba’. And it is clear that the connections with large weights from the input unit for ‘b’ to one hidden unit, and from that hidden unit to the output unit for /B/ make up a component mechanism that is a causal common factor in the five ‘b’-to-/B/ transitions.

We can also give a simple example of a network that uses the same input and output coding schemes but does not have implicit letter-sound rules. Figure 4 shows a network in which one hidden unit is dedicated to each of the twenty five letter strings in the task domain. The input unit for ‘b’ has connections of strength 0.5 leading to five hidden units (for ‘ba’, ‘be’, ..., ‘bu’), and the input unit for ‘a’ likewise has connections of strength 0.5 to five hidden units, including the unit for ‘ba’. The hidden units themselves are simple threshold units (and this non-linearity is an important aspect of the example). The threshold

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Strong</th>
<th>Intermediate</th>
<th>Weak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule explicit</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Modestly modular</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Madly modular</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Knowledge of rules in the three configurations
is set to 0.75, so that a hidden unit is switched on if and only if both the corresponding input units are active. Each hidden unit passes activation to the two corresponding output units. The two input-hidden connections and two hidden-output connections that are shown in bold explain the input-output transition for ‘ba’. But now, there is nothing in common between these connections and the four connections that are involved in the transition for ‘be’.

The first notion of implicit rules thus permits us to ask whether a network (with or without hidden units) embodies spelling-sound rules; to that extent it is just the kind of notion that Seidenberg (1989) invited us to develop. Much more work would need to be done, of course, before we could sensibly pose the question in cases where the hidden units are neither linear units nor threshold units but use the standard sigmoid activation function, or cases where complicated schemes of distributed input or output representations are employed.

The first notion of implicit rules also allows that a rule can be embodied in the weight on an individual connection, just as Bates and Elman (1993) said. What we were aiming at, though, was a notion that would also permit one and the same rule to be embodied in different configurations of weights in different networks. It should now be clear that the first notion of implicit rules does meet this requirement. The diagnostic question concerns the structure of causal explanations of input-output transitions rather than the fine-grained detail of how each transition is achieved. Thus, a connectionist network with hidden units, a network without hidden units, and a classical symbol manipulation device can have the same rules implicit in them.

2.4 Two consequences

In the next section, I shall introduce a second notion of implicit rules. But to conclude this present section, I shall note two consequences of the first notion of implicit rules.

2.4.1 Systematicity and syntax

One consequence follows from the fact that causal systematicity of process imposes requirements upon the causal properties of input states. Suppose that we define a syntactic (or formal) property of a representational state to be (i) a physical property of the state that is both (ii) systematically related to the state’s semantic properties and (iii) a determinant of the state’s causal consequences (Fodor, 1987, pp. 16-21). Then it is possible to show that implicit rules in a system require a measure of syntactic articulation in the system’s input states (Davies, 1989, 1991). In particular, the presence of implicit rules about the pronunciation of individual letters requires that the input states that represent letter strings should have internal syntactic structure corresponding to the individual letters in the represented string. This requirement is met in the modestly modular network (Figure 1) by the articulation within the patterns of activation that repre-
sent the letter strings. All the input states that represent strings ‘b ’, for example, have the property of involving activation of the first unit in the first pool. This is surely a physical property of those input states—(i). It is a property that is correlated with a semantic property, namely that the state represents a string beginning with ‘b’—(ii). And it is a property which, in the presence of the connections between input and output units in the network, is a determinant of the input state’s causal consequences—(iii). There is, of course, no such syntactic articulation in the input states that represent letter strings in the madly modular network (Figure 2). The input states of the network in Figure 4 are no less articulated than those in the modestly modular network, but still the spelling-sound rules are not implicit there. In essence, the structure that is present in the input states is thrown away at the hidden unit layer. Syntactic articulation in input states is a necessary, but not a sufficient, condition for causal systematicity of process in input-output transitions.

2.4.2 Formal theories and causal processes

The other consequence of the first notion of implicit rules concerns formal theories of a cognitive task. The rules that characterize a task, such as the reading aloud task, can be stated by the axioms of a formal theory. The particular consequences of those rules, such as statements of the pronunciations of particular letter strings, can be derived from those axioms as theorems of the theory. Furthermore, it is possible to specify a canonical proof procedure that is to be followed in deriving those theorems from the axioms. An individual axiom can then figure as a derivational common factor in the (canonical) proofs of several theorems. For example, the axiom stating the letter-sound rule for ‘b’ would figure as a derivational common factor in the proofs of the five theorems that state the pronunciations of the strings ‘ba’, ‘be’,..., ‘bu’.

There is then a very evident parallel between formal theories and causal processes. If the rules stated by the axioms of a formal theory of a task are implicit in a processing system, then the causal structure of the processing in the system mirrors, or follows the contours of, the derivational structure of the (canonical) proofs in the formal theory. Where an axiom functions as a derivational common factor in the proofs in the theory that concern certain items in the task domain, a component processor functions as a causal common factor in the processing in the system that begins from input states representing those items.

3. The second notion of implicit rules

The notion of implicit rules to be introduced in this section imposes fewer requirements upon the causal processes taking place inside a system. The core of this second notion is the idea of a rule that could be extracted or derived from information that is present in the system.
3.1 What is present and what can be derived

In a discussion of explicit, implicit, and tacit representation, Daniel Dennett suggests (1983, p. 216):

[...]et us have it that for information to be represented implicitly, we shall mean that it is implied logically by something that is stored explicitly.

Applied to the case of a rule, Dennett’s account would have the result that an implicit rule is one that is logically implied by information that is explicitly represented in the system. The second notion of implicit rule that I am introducing is a generalization or relativisation of that idea. Relative to any given notion of information being present in a system—of which, information being explicitly represented is one—a rule is implicit in the system if it can be derived from information that is already present in the system. This is not to say that the system itself contains mechanisms for performing these logical derivations.

There is scope for a second relativisation in this notion of an implicit rule. For we might consider the use of different resources for deriving rules from a body of information. Dennett speaks of what is ‘logically implied’, but we might also permit broadly inductive methods of rule derivation (extraction), for example. However, this second relativisation will not concern us here.

Suppose now that, instead of taking the notion of information that is explicitly represented as our baseline, we make use of the intuitive idea that the information that ‘ba’ is pronounced /BA/, and the information that ‘be’ is pronounced /BE/, and so on, is present in all three of the systems that we considered in Section 2.2—the rule-explicit system, the modestly modular network, and the madly modular network. As well as being an intuitive idea, this is an idea that is underwritten by the first notion of implicit rules. The twenty-five trivial (one-case) rules are implicit in all three systems according to that first notion. Then, relative to that starting point, the ten letter-sound rules may be implicit (according to the second notion of implicit rules) even in a madly modular network.

To see this, consider two formal theories of some cognitive task. Our simplified reading aloud task can serve as an example. The first formal theory, T₁, is a structured theory with axioms stating the ten letter-sound rules. The second formal theory, T₂, is a ‘listiform’ theory. Its axioms simply state the twenty-five one-case rules for pronouncing the twenty-five items in the task domain. According to the first notion of implicit rules, the rules of the structured theory, T₁, are implicit in the modestly modular system (Figure 1) but not in the madly modular system (Figure 2). But, as we have just noted, the one-case rules of the listiform theory, T₂, are implicit in both the modestly and the madly modular systems.

In certain cases, two theories related as T₁ and T₂ are logically equivalent. The axioms of T₂ are theorems of T₁ in any case. But, where the two theories are
logically equivalent, the structured rules can actually be derived from the one-case rules of the listiform theory. This is the case with the two theories of the reading aloud task. Or rather—to be properly delicate—this is the case provided that T₂ is augmented by a theory of the task domain that says that there are just the five consonants that can occur in the first position in a string and just the five vowels that can occur in the second position. So—with that proviso—the ten letter-sound rules of T₁ appear to be implicit (according to the second notion) even in the madly modular system. Since the one-case rules that we are taking to be present in the system are themselves implicit (according to the first notion) rather than explicitly represented, we might say that the ten letter-sound rules are doubly implicit in the madly modular network.

3.2 Articulation in the input representations

The conclusion that the letter-sound rules are doubly implicit in the madly modular network (Figure 2) is almost correct, but we have to face up to a complication that will be important in Section 4.

I said that the letter-sound rules can be derived from statements about the pronunciations of individual letter strings. Thus, for example, the rule for the letter ‘b’, which we can state as:

any two-letter string ‘b_’ (that is, any string that begins with the letter ‘b’) is pronounced /B_/ (that is, has a pronunciation that begins with the sound /B/)

is to be derived from five of the axioms of the listiform theory:

‘ba’ is pronounced /BA/
‘be’ is pronounced /BE/
‘bi’ is pronounced /BI/
‘bo’ is pronounced /BO/
‘bu’ is pronounced /BU/

along with the background assumption that there are just the five vowels ‘a’, ‘e’, ‘i’, ‘o’, ‘u’ that can go together with ‘b’ to make up a two-letter string. But now we should consider carefully the nature of the terms that are used in the theory to designate the two-letter strings in the task domain—the term “‘ba’” for example. It is crucial that these terms should be structural descriptions of letter strings and not just unstructured names of those strings.

If the term “‘ba’” is just a label for the two letter string made up of ‘b’ followed by ‘a’, then the axioms of the listiform theory are tantamount to:

#1 is pronounced /BA/
#2 is pronounced /BE/
and so on. And from these axioms, absolutely nothing about letter strings beginning with the letter ‘b’ follows logically, since these axioms do not tell us, for example, that string #1 begins with ‘b’. What is needed is that the axioms of the listiform theory should state that:

the two-letter string made up of ‘b’ followed by ‘a’ is pronounced /BA/
the two-letter string made up of ‘b’ followed by ‘e’ is pronounced /BE/

and so on. From such axioms we certainly can derive conclusions about letter strings beginning with ‘b’.

We said that what is stated by the axioms of the listiform theory is information that is present even in the madly modular network. But now we see that more care is required. Are the rules stated by the axioms of the structured theory implicit in that network (according to the second notion of implicit rules)? The answer is highly sensitive to whether the information present in the madly modular network is the information that the two-letter string made up of ‘b’ followed by ‘a’ is pronounced /BA/ or merely the information that string #1 is pronounced /BA/. Does the madly modular network contain any information about the internal structure of the twenty five twp-letter strings in the task domain?

To this question it is only possible to give an intuitive answer, since we are making use of an intuitive notion of information being present in a system—a notion only partially underpinned by the first notion of implicit rules. But the intuitive answer is surely that the madly modular network as it stands does not contain information about the structure of the objects in its task domain. It is difficult to see how a system can contain the information that string #1 has ‘b’ as its first letter, if the system has no way of representing the letter ‘b’.

One way—arguably the most natural way, though certainly not the only conceivable way—of making it more plausible that the madly modular network contains the information from which the ten letter-sound rules can be derived is to incorporate some structure or articulation into the input representations that it uses. We can add extra input units to represent the constituent letters of the strings, make appropriate connections between those letter units and the input units that represent the letter strings themselves, and set appropriate activation thresholds for the letter string units. Then, even if the network’s way of representing the string ‘ba’ (activation at the letter string input unit) is just an unstructured label, still we can plausibly credit the network with the information that the string thus labelled contains ‘b’ and ‘a’. Consequently, we really can say that the letter-sound rules are implicit in this network (according to the second notion of implicit rules). Figure 5 depicts the madly modular network augmented in this way. (Similar augmentation is provided at the output end of the network.)

It should be clear that, despite our tinkering with the input and output representations, this network still does not license an affirmative answer to the diagnostic question for the first notion of implicit rules. The articulation in the input representations is not utilized in the transition from the letter string input
unit to the phoneme string output unit. The connection whose presence is crucial to the explanation of the ‘ba’-to-/BA/ transition is still explanatorily irrelevant to the ‘be’-to-/BE/ transition, for example.

In preparation for the next section, there are two points to notice about the network in Figure 5. The first point is that the scheme of input representation is a simple case of tensor product coding. The units for the letter strings are binding units, while the separate units for the consonants and vowels in first and second position in the strings are filler and role units (see e.g. Smolensky 1987; 1990, pp. 147-53).

The second point is that once a binding unit—say, the unit for the string ‘ba’—is activated, the presence of the filler and role units—the units for ‘b’ and ‘a’—is causally irrelevant to the subsequent feed-forward processing. The causal consequences of activation at the ‘ba’ unit are exactly the same whether that unit is switched on by activation passed from the ‘b’ and ‘a’ units or is clamped on by external intervention.

3.3 Summary: The two notions

In this section and the previous one, I have introduced two notions of implicit rules. The first notion is defined in terms of causally systematic processes (or causal common factors in processing), and is genuinely intermediate between the strong and weak notions of knowledge of rules discussed in Section 1. Implicit rules do not need to be encoded in syntactically structured representations, but there is a connection with syntactic structure nevertheless: implicit rules require syntactic articulation in the system’s input states. The second notion is defined in terms of the derivation of a rule from information that is present in the system. This notion imposes fewer requirements on the causal processes that take place within the system, as witness the fact that implicit rules may be present according to this second notion where they are absent according to the first notion. But the second notion of implicit rules arguably requires some articulation in the system’s input representations, such as the structure provided by tensor product coding. In the final section, I shall use these two notions to shed some light upon a dispute about structure-sensitive processes.

4. Structure-sensitive processes

There are many things at issue between the advocates and opponents of connectionism. In the debate between Fodor (and Pylyshyn and McLaughlin) and Smolensky, a great deal seems to rest upon the question whether the processing in connectionist networks is structure-sensitive. Fodor claims that mental (generally, psychological) processes are structure-sensitive, and that connectionist networks are not good models of mental processes since processing in networks is not structure-sensitive. Clearly, there are two ways for an advocate of connectionism to respond to this challenge. He can deny that mental processes
are structure-sensitive, or he can insist that processing in connectionist networks can be structure-sensitive. On the face of it, Smolensky takes the second course, but his claim to have delivered connectionist structure-sensitivity by way of tensor product coding has been greeted with incredulity by Fodor (1991, p. 279):

As for Smolensky's 'tensor product' defense of connectionism... As far as I can tell, the argument has gone like this: Fodor and Pylyshyn claimed that you can't produce a connectionist theory of systematcity. Smolensky then replied by not producing a connectionist theory of systematcity. Who could have foreseen so cunning a rejoinder?

My suggestion in this final section is that the parties to this debate are talking past each other to some extent, since they are using different notions of structure-sensitivity. What Smolensky is offering is significant in its own right; but it is not what Fodor was asking for.

4.1 Fodor's notion of structure-sensitive processes

What is the notion of structure-sensitive processes that Fodor is working with? To begin with, a psychological process involves operations over representations. An information processing system can, obviously enough, be described in terms of the information that is being processed. A system progresses from having the information that a certain presented item is (a token of) the letter string 'ba' to having the information that the presented item is to be pronounced /BA/. The basic idea of information processing psychology is that this progression occurs because of two things. There is some state of the system—whether an occurrence of a formula in some linguistic format, or a pattern of activation over units—that represents the letter string 'ba'. And there is some operation—whether symbol manipulation or passing activation forward along weighted connections—that gives rise to a second representational state, this time representing the pronunciation /BA/. Thus Fodor says (1987, p. 145):

If you think of a mental process—extensionally, as it were—as a sequence of mental states each specified with reference to its intentional content, then mental representations provide a mechanism for the construction of these sequences; they allow you to get, in a mechanical way, from one such state to the next by performing operations on the representations.

This idea of psychological processes as achieved by operations performed upon representations applies to all the models that we have considered, whether rule-explicit or rule-implicit, modestly or madly modular, with or without hidden units. Structure-sensitive psychological processes require something more (Fodor and Pylyshyn, 1988, p. 13):

Because Classical mental representations have combinatorial structure, it is
possible for Classical mental operations to apply to them by reference to their form. The result is that a paradigmatic Classical mental process operates upon any mental representation that satisfies a given structural description, and transforms it into a mental representation that satisfies another structural description.

In a structure-sensitive process, the representations over which the operations are performed have syntactic structure, and the operations make use of that syntactic structure.

Suppose that the system's representations of the strings 'ba', 'be', 'bu' have syntactic structure. In the case of the representation of 'ba', this means that there are two physical properties—(i), that are correlated with the semantic properties of designating a string containing 'b' and designating a string containing 'a'—(ii), and that are determinants of the causal consequences of the occurrence of that representational state within the system—(iii). Considering the five representations of the 'b' strings, we can see that they have a physical and causal property in common; namely, the physical property corresponding to the semantic property that they share (they all designate strings containing 'b'). This common causal property of the five input representations certainly invites the application of a single operation in order to make a systematic contribution to the production of output representations of the pronunciations of the five strings. Syntactically structured input representations lend themselves perfectly to causally systematic input-output transition processes, although (Figure 4) structure in the input representations does not guarantee causally systematic processing thereafter.

I have already said—though I did not give the argument in this paper (see Davies 1989, 1991)—that causally systematic processes require syntactic structure in a system's input representations. And we have just seen that the syntactic properties of representations are made for—they go hand in glove with—causally systematic processes. In short, Fodor's notion of a structure-sensitive process is essentially the same as the idea of a process in which a rule is implicit according to the first notion of implicit rules.

4.2 Smolensky's notion of structure-sensitive processes

One of the examples that Fodor and Pylyshyn (1988) give of a process that would certainly be treated as structure-sensitive within a Classical framework is the process of logical inference, such as the inference from P&Q to P. Smolensky addresses this case in terms of tensor product coding (1991a, p. 216):

I have talked so far mostly about representations and little about processing. If we are interested, as [Fodor and Pylyshyn] are, in inferences such as that from P&Q to P, it turns out that with tensor product representations, this operation can be achieved by a simple linear transformation upon these representational vectors,... . Not only can this structure-sensitive process be achieved by
connectionist mechanisms on connectionist representations, but it can be achieved through the simplest of all connectionist operations...

In tensor product coding, patterns of activation over the role units and the filler units produce a pattern of activation over the binding units. It is a general feature of this technique that—subject to certain conditions—given a pattern of activation over the binding units and the original pattern of activation over the role units it is possible to recover the original pattern of activation over the filler units. This is called unbinding (the filler from the role), and it could be used as Smolensky says to recover a representation of the left conjunct from a tensor product representation of a conjunction.

Smolensky goes on to say (1991a, p. 217):

[T]he agenda for connectionism should [be]...to develop formal analysis of vectorial representations of complex structures and operations on those structures that are sufficiently structure-sensitive to do the required work. This is exactly the kind of research that, for example, tensor product representations are being used to support.

But Fodor can reply that, however things may be with tasks that can be treated as cases of unbinding, tensor product coding does not, in general, address the issue of structure-sensitivity as he conceives it.

The reason is that, for Fodor, structure-sensitivity is a causal matter—an operation engages causally with the syntactic or formal properties of the representations upon which it is performed. But, if tensor product coding is used for the input representations in some system, and if the forward connections run from the binding units, then the backward connections from the binding units to the filler and role units are not causally relevant to the operations that produce the system’s outputs.

Consider once again the input representation of the string ‘ba’ in the network in Figure 5. Activation at the ‘ba’ letter string input unit has the properties of being partly caused by activation at the ‘b’ unit and being partly caused by activation at the ‘a’ unit. These physical properties of the input representation are correlated with its semantics: it designates a string containing ‘b’ and ‘a’. But, as we noted towards the end of Section 3, these backward-looking causal properties are not themselves determinants of the causal consequences of the activation of the ‘ba’ unit. In fact, the causal irrelevance for feed-forward processing of the activation at the filler and role units can be dramatized by noting that in some examples of tensor product coding it is only the binding units that are really present in the network; the filler and role units are merely virtual. All this is in sharp contrast to the input representation of the string ‘ba’ in the modestly modular network in Figure 1. In that case, the pattern of activation that represents the string ‘ba’ has the properties of being partly constituted by activation at the ‘b’ unit and being partly constituted by activation at the ‘a’ unit.
These properties are correlated with the semantics of the input representation, but they are also clearly determinants of the causal consequences of the overall pattern of activation which they jointly constitute. Thus Fodor and McLaughlin (1990, p. 200):

The relevant question is whether tensor product representations...have the kind of constituent structure to which an explanation of systematicity might appeal. But we have already seen the answer to this question: the constituents of complex activity vectors typically aren't 'there', so if the causal consequences of tokening a complex vector are sensitive to its constituent structure, that's a miracle.

4.3 Complications and confounds

There are similarities between the network in Figure 5 and the one in Figure 4. We could turn the network in Figure 5 into the network in Figure 4 by just collapsing the phoneme string output units with the letter string input units. In neither network are the letter-sound rules implicit. But we are using the two networks to make different points.

The network in Figure 4 shows that syntactic structure in input representations (patterns of activation over letter units) might not be utilized for causally systematic processing. The network in Figure 5 shows that input representations (activation of letter string units) may have physical properties that correlate with semantics, without those physical properties being determinants of the causal consequences of those input states. This is generally illustrated by the backward-looking causal relational properties of the activation of binding units.

There are points at which Smolensky seems to embrace the thought that he is not offering the kind of structure-sensitivity that Fodor is asking for. For example (Smolensky, 1991a, p. 222): 'Are the vector constituents in physical and connectionist systems causally efficacious? It would appear not...' However, there are two confounds that make it difficult to assess the import of this apparent concession.

The first confound is that, in this brief remark, Smolensky is not, in fact, talking about the 'constituents' in tensor product representations. That is, he is not talking about the patterns of activation over filler and role units that give rise to a pattern of activation over binding units. Instead, he is talking about components in a vector sum. It is a familiar point that, in connectionist theory, activation over a pool of n units is often represented as an n-place vector. So, suppose that we have a pattern of activation, represented in that way. Then, since there is no unique way to decompose a vector into components, there will be many different ways to see the pattern of activation as a sum of sub-patterns. Smolensky's point is then that the component sub-patterns are not causally efficacious, even though decomposing activation vectors in certain ways may be crucial 'in order to understand and explain the regularities in the network's
behavior’ (1991a, p. 221). Whether or not this claim about causal efficacy is correct (in fact, it arguably is not), it is clear that Smolensky’s remark does not straightforwardly address Fodor and McLaughlin’s concerns.

The second confound concerns a claim that Smolensky appears to make (p. 222), namely that it is safe to make his concession about the lack of causal efficacy in the case of connectionist constituent structure, since there is arguably no more efficacy of constituent structure in the Classical case. The problem is that the argument for this claim depends upon an analogy with software (ibid.):

When we write a Lisp program, are the symbolic structures we think in terms of causally efficacious in the operation of the computer that runs the program?

The answer to this question (which we may suppose to be negative) is not strictly relevant, since Fodor and McLaughlin are concerned with the syntactic properties of the representations that are actually implicated in the operation of an information processing system, not the syntactic properties of a programme that may be removed from that operation by many steps of compiling.

Smolensky’s explicit mention of the question of causal efficacy seems only to muddy the waters. It really is not quite clear what he takes the requirement of structure-sensitivity to amount to. But we can make a sympathetic proposal for reducing the mutual incomprehension that characterizes this debate.

4.4 Implicit rules and structure-sensitive processes

Having introduced two notions of implicit rules, we have already observed that Fodor’s notion of structure-sensitive processes corresponds to the first notion of implicit rule. (‘F’ for ‘Fodor’, and for ‘first’.) The F-notion of implicit rules requires causally systematic processing—as assessed by the diagnostic question of Section 2.1—and this in turn requires syntactic articulation in the system’s input representations.

We can introduce another notion of structure-sensitive processes corresponding to the second notion of implicit rules. The sympathetic proposal is simply that this is the notion of structure-sensitive processes that Smolensky is employing, whether intentionally or not. (‘S’ for ‘Smolensky’, and for ‘second’.) The S-notion of implicit rules arguably requires some articulation in input representations, for example, the kind of articulation that is provided by tensor product coding (Section 3.2); hence Smolensky’s focus upon this style of input representation. But nothing that is required by the S-notion guarantees causally systematic processing (as the network in Figure 5 illustrates); hence Fodor’s remark about ‘so cunning a rejoinder’.

Smolensky reckons himself to have delivered; but Fodor pronounces himself unsatisfied. No wonder: Fodor is asking for F-structure-sensitivity, while Smolensky is offering S-structure-sensitivity. That, by my lights, is the source of the mutual incomprehension. But to diagnose is not yet to resolve.
Armed with our distinctions, we can see that the debate should be reconfigured. When it comes to structure-sensitivity, the agenda for connectionist research needs to have two aspects. On the one hand, it needs to develop tensor product representations as inputs to S-structure-sensitive processes. On the other hand, it also needs to explore what limitations there may be upon F-structure-sensitive processes in networks. Certainly, F-structure-sensitivity is not ruled out altogether; the networks in Figures 1 and 3 illustrate that. The significance of the results of this research will then be conditioned by findings from (human) psychological research that addresses the question whether mental processes are really F-structure-sensitive or only S-structure-sensitive.

Putting this in terms of implicit rules, we can say that the research agenda is to discover whether connectionist networks and human mind/brains embody implicit rules. But to ask that question productively, we must distinguish two notions of implicit rules.

References


Davies, M. 1990a: Knowledge of Rules in Connectionist Networks. Intellectica nos.9-10: D. Memmi and Y.M. Visetti (eds), Modèles Connexionnistes, 81-126


Seidenberg, M. S. and McClelland, J.L. 1989: A Distributed, Developmental Model of Word Recognition and Naming. _Psychological Review_, 96, 523-68
Figure 1: The modestly modular configuration
Figure 2: The madly modular configuration

- Input units
  - 'ha'
  - 'be'
  - 'bi'
  - 'bo'
  - 'ku'

- Output units
  - /BA/
  - /BE/
  - /BI/
  - /BO/
  - /KU/
Figure 3: A hidden unit network with implicit letter-sound rules (not all connections are shown)
Figure 4: A hidden unit network without implicit spelling-sound rules (not all connections are shown)
Two Notions of Implicit Rules

Figure 5: The madly modular configuration with articulation in the input (and output) representations