4

Developmental Transitions

In this chapter, we consider the second of the two major developmental issues, namely how children effect transitions from one state or stage of knowledge to the next. Although issues of transition are widely considered to be of extremely high priority, it has been reported that many developmental researchers have not pursued these issues because they considered problems of transition to be too difficult (Flavell, 1984; Newell, 1990). This inherent difficulty, combined with nativist arguments about language acquisition and contemporary research discovering various competencies in young infants, has driven work on developmental transition into the background in recent years. Because of the abilities of networks to learn and self-organize, neural-network approaches to development have driven a revival of interest in mechanisms of psychological transition (Bates & Elman, 1993; Elman et al., 1996). In addition, mainstream psychological work on developmental transition has continued in at least a few laboratories (e.g., Case, 1985; Siegler & Jenkins, 1989; Sternberg, 1984).

This chapter opens with a quick overview of both classical and contemporary psychological proposals for transition mechanisms and a computational interpretation of these in terms of connectionist and rule-based algorithms. It may be possible to achieve a theoretical integration of these various psychological proposals by considering them from a computational perspective, particularly from the perspective of a generative algorithm such as cascade-correlation (as introduced in chapter 2). Then, as a prelude to a comparative modeling of three case studies of transition, we present the basics of a leading rule-learning algorithm called C4.5. After the three case studies (on balance-scale, conservation,

and habituation-of-attention phenomena), we consider four basic theoretical issues about transition: How can anything genuinely new be learned? What is the relation between learning and development? How might innate determinants operate? And finally, when is it appropriate to use generative versus static networks in modeling psychological development?

Proposed Transition Mechanisms

My coverage of influential proposals for transition mechanisms include both classical Piagetian theory and a variety of more contemporary ideas.

Piaget's theory of transition

Because Piaget's theory is not in much favor these days, a legitimate question would be, why bother with it? The reason for bothering is that, despite the critiques it has received, Piaget's theory remains the single most serious, ambitious, and influential attempt to describe and explain cognitive development (Case, 1999; Miller, 1989; Siegler, 1998). Furthermore, detailing Piaget's theory and giving it a computational interpretation may help to deal with one of the most devastating criticisms—that the theory is hopelessly vague, particularly when it comes to transition mechanisms (Boden, 1982; Klahr, 1982). Finally, giving Piaget's theory a computational interpretation may help some readers who are familiar with Piagetian psychology to better understand and appreciate the computational approach.

Earlier attempts to discuss Piaget in neural-network terms (McClelland, 1995; Shultz, Schmidt, Buckingham, & Mareschal, 1995) did not quite succeed, principally because they focused on Piaget's older ideas on assimilation and accommodation, ignoring his more recent notion of abstraction. The twin processes of assimilation and accommodation may be relatively easier to understand, and many developmental researchers have interpreted them as Piaget's definitive treatment of transition (e.g., Gallager & Reid, 1981; Klahr, 1982; Nersessian, 1998; Siegler, 1998).

Assimilation refers to distortion of information from the environment to fit the child's current cognitive system (Piaget, 1980). *Accommoda-*

tion is the contrasting process in which cognition is adapted to external information. Assimilation and accommodation can be viewed as two different sides of the same coin, because assimilation typically prompts accommodation and accommodation, in turn, improves future assimilation. *Equilibration* is the process that maintains balance between assimilation and accommodation, ensuring that sufficient accommodation occurs to enable successful assimilation. The process of equilibration produces a state of *equilibrium*. However, such an equilibrium is only temporary if it occurs within an inadequate cognitive structure. Repeated conflict between the child's cognitive achievement and disconfirming environmental feedback eventually prompts some kind of cognitive reorganization.

Such reorganization occurs through the process of *reflective abstraction*. Reflective abstraction consists of two distinct processes: *reflecting*, which is projection to a higher level of what is occurring at a lower level, and *reflection*, which is reorganization at the higher level (Piaget, 1980).[1]

It is possible, and perhaps instructive, to map these Piagetian transition mechanisms onto computational algorithms used to simulate cognitive development. Perhaps the clearest and fullest such mapping can be done to the cascade-correlation algorithm, introduced in chapter 2.[2] In this mapping, which is summarized in table 4.1, *assimilation* corresponds to the forward propagation of inputs through a network to the output units, thus generating a response to a stimulus pattern. This forward propagation occurs without any network modification, and so could

Table 4.1
Mapping Piagetian theory to computational features of cascade-correlation

Piaget	Cascade-correlation
Assimilation	Forward propagation of inputs
Accommodation	Output-weight training
Equilibration	Error reduction
Equilibrium	Error stagnation
Conflict	Error
Reflective abstraction	Hidden-unit recruitment
Reflecting	Input-weight training
Reflection	Output-weight training (after recruitment)

represent assimilation of an input pattern to a network's current structure (topology) and knowledge (connection weights). *Accommodation*, in turn, can be mapped to connection-weight adjustment, as it occurs in the output phase of cascade-correlation learning. Here a network changes in response to discrepancy between what it produces (on an output activation vector) and what it sees in the environment (in a target activation vector). This implements a process of equilibration in which error is reduced as weights are adjusted. A state of equilibrium, representing a balance between assimilation and accommodation, is reached when weights stop changing. This could be either a permanent equilibrium, if error can be reduced to some minimum value, or a temporary equilibrium, representing the best a system can do with its current computational power. Note that any such changes brought about by weight adjustment are quantitative changes made within the network's current topology. The numbers on the weights change, but the network's processing is not qualitatively different than before. Weight adjustments are not qualitative changes that create new structures.

More substantial qualitative changes, corresponding to *reflective abstraction*, occur as new hidden units are recruited into the network. In cascade-correlation, this occurs in two phases. First is the input phase, in which current input- and hidden-unit activations are projected onto a new level, that of candidate hidden units. This might correspond to *reflecting* in Piaget's terms. When the system finds the hidden unit best correlating with network error, it is installed into the network, downstream of the input and hidden units that project onto it, with weights frozen at their current, recently trained levels to ensure that the unit's ability to track network error remains unchanged. Then the network reverts back to an output phase in which it tries to incorporate the newly achieved representations of the recruited unit into a better overall network solution. This, of course, could correspond to Piaget's notion of *reflection*.

One possible difference between this computational reinterpretation and Piaget's original theory is that output-phase weight training can be identified with both *accommodation* and *reflection*, at least after the first output phase. Both of these processes represent attempts to solve the problem posed by the training patterns within the current computational structure, a structure that is changing with experience and development.

Neither Piagetian psychology nor cascade-correlation requires or depends upon this mapping of one theory onto the other. Each is a source of theoretical ideas that might survive or fall based on psychological evidence. But linking the two approaches in this way could promote understanding of one by practitioners of the other and perhaps lead to new insights about psychological development.

Cascade-correlation was not in any way designed to mimic Piaget's theory of transition. Indeed, it was designed instead to improve on the back-propagation algorithm and perhaps be applied to various engineering problems that require a learning approach. Nor was our choice of cascade-correlation for simulating psychological development meant to merely implement Piagetian theory in a computational fashion. Indeed, the mapping presented here was achieved only recently (Sirois & Shultz, 2000) and after previous false starts. It represents a fortuitous and perhaps interesting way to relate the most influential psychological theory of development to a principal neural-network algorithm for simulating cognitive development.

How might Piaget's transition mechanism be mapped to other simulation methods? Piaget's views on transition were mapped to back-propagation learning by McClelland (1989). As I just did, McClelland likened accommodation to weight adjustment and assimilation to processing without any such adjustment. In rule-based learning systems, assimilation could be construed as rule firing, and reflective abstraction as rule learning. It is unclear how the full range of Piaget's ideas on transition can be accounted for in these alternate approaches. Connection weight adjustment seems destined to create only quantitative change within a constant structure. Rule learning, on the other hand, seems to involve only qualitative change, although quantitative changes in rules can be implemented in terms of their priority, utility, or certainty of conclusions, and this could correspond to Piagetian accommodation.

Contemporary views of transition

In comparison to Piaget's rich and somewhat exotic formulation, contemporary views of transition mechanisms can seem a bit pedestrian. A widely held view from the information-processing perspective that has dominated cognitive psychology for the past 40 years is that children develop by learning, more specifically, by *altering the contents of long-*

term memory (LTM). This could mean learning rules, for theorists who believe that human knowledge is coded in rules (e.g., Newell, 1990), or it could mean learning other, still somewhat underspecified kinds of declarative knowledge (e.g., Chi, 1978). There is considerable evidence that not only do children acquire knowledge in many domains but also their knowledge determines how and how much they can learn. For example, children who are experts in chess remember more about chess positions to which they are briefly exposed than do adults who are not experts in chess (Chi, 1978). These child experts exhibit a case of LTM knowledge aiding working memory (WM). Despite such occasional cases in which children can acquire deep expertise, the general idea is that very young children are universal novices, and that they develop by learning expertise in a variety of domains. On this view, if we could understand how children build and modify their LTMs, we would identify a transition mechanism for cognitive development. *Acquisition of strategies* for information processing (Siegler & Jenkins, 1989) could also be considered in terms of changes in LTM.

Other proposals, also from the information-processing approach, emphasize working memory (WM), rather than LTM, as a source of cognitive development. There has been a prolonged debate over the possibility that children *gain in WM capacity* as they mature. A number of researchers have proposed that such capacity changes are responsible for much of the improved cognitive performance seen in children with advancing age (Case, 1985; Fischer, 1980; Pascual-Leone, 1970). For example, Case (1985) argued that children can handle a single piece of information at 3–4 years of age, up to two pieces of information at 4–5 years of age, three at 7–8 years, and four at 9–10 years of age. On a rule-based theory of cognitive processing, it is a great mystery how children ever solve any realistic-sized problems if that is all the information that they can handle in WM. Recall, for example, the extensive WM requirements of a production-system treatment of the notion of paternal grandfather discussed in chapter 3. That was a very simple problem of inferring who was the paternal grandfather of whom in a small number of people. Admittedly, the example used a memory-hungry backward-reasoning program, but even a more modest forward-reasoning program would require far more WM capacity than Case's limits.

In general, assessing WM capacity is a huge problem because of the well-known tendency for such capacity to expand with familiar material, that is, information stored in LTM. The limit is on chunks of information, which can be further broken down into subchunks and so on, using information about these chunks that is stored in LTM (Miller, 1956). For example, it is easier to remember alphabetic letters embedded in familiar words than presented separately, and easier to remember words in a meaningful sentence than words presented randomly. Thus, because of the pervasive influence of LTM knowledge on WM processing, the hypothesis that expansion of WM is a transition mechanism in cognitive development may be essentially untestable.

Another potential problem for the expanding-WM theory of change is that the cause of any such expansion may not be easy to identify. It has sometimes been attributed to connectivity changes in the frontal lobes (Case, 1992) and other times to automatization of mental operations (Case, 1985). Basically, as mental skills become well-practiced (automatized), this frees up WM for other operations, providing a kind of functional increase in WM capacity. Consequently, *automatization* is often proposed as yet another, independent source of cognitive development (Siegler, 1998).

A somewhat related idea is that cognitive development is a function of *increases in processing speed* of both WM manipulations and retrieval of information from LTM (Hale, 1990; Kail, 1986, 1988, 1991). For example, 5-year-olds can take up to three times as long to execute the same mental operation as 14-year-olds. Like differences in WM capacity, differences in processing speed could account for large differences in cognitive performance. Such increases could, for example, counteract the rapid decay of information from WM.

As in the case of changes in WM capacity, there are multiple causal possibilities here. It might be that processing speed increases with brain maturation, as suggested by evidence that children who are more physically mature are mentally faster than children who are less physically mature (Eaton & Ritchot, 1995). Or it might be that older children and adults are faster processors because of more practice at cognition, as suggested by evidence that mean processing speeds of younger and older individuals can be plotted on the same learning curve (Kail & Park, 1990).

Yet another modern proposal for a transition mechanism is *encoding*, that is, identifying the most relevant and informative features of stimuli and using these features to form mental representations of the stimuli. Encoding has figured in several influential theories of transition (e.g., Klahr & Wallace, 1976; Sternberg, 1985) as well as being an essential feature of memory theories (e.g., Brainerd & Reyna, 1990). Encoding is also likely to be involved in the effect of knowledge on new learning. Basically, knowledge can focus a learner's attention on the most important features of the material to be learned.

Some of the most convincing evidence on encoding and learning involves the balance-scale problem, introduced in chapter 3. Siegler (1976) provided various kinds of balance-scale feedback to 5- and 8-year-olds who were diagnosed at stage 1 of balance-scale performance. Children of both ages who received feedback on distance problems (which are typically failed in stage 1) usually progressed to stage 2. Also, children of both ages who received feedback only on weight problems (correctly solved in stage 1) continued to perform at stage 1. However, only 8-year-olds profited from feedback on the more difficult, conflict problems, often moving to stage 3, which is characterized by sensitivity to both weight and distance information.

These training effects were simulated in a back-propagation model in which age was manipulated by the amount of training (McClelland & Jenkins, 1991). A network at epoch 20 represented 5-year-olds, and a network at epoch 40 represented 8-year-olds. When trained on additional conflict problems, the "older" network quickly progressed to stage 2, but the "younger" network did not. Examination of connection weights provided a mechanistic description of how the older network was more ready to learn than the younger network. Essentially, the younger network was not yet effectively encoding distance information, as indicated by a small range of connection weights. Consequently, this network could not benefit from conflict problems in which distance information was critically important. In contrast, the older network did encode distance information, as indicated by a relatively large range of connection weights, albeit too weakly to actually produce stage 2 behavior. Thus, a small amount of additional relevant experience, in the form of conflict problems, allowed the older network to progress to stage

2. This simulation goes a long way to demystify the notions of readiness to learn and conceptual precursors.

To test the idea that encoding difficulties prevented 5-year-olds from benefiting from the conflict-problem feedback, Siegler assessed the encoding skills of 5- and 8-year-olds diagnosed at stage 1 by asking them to reconstruct balance-scale configurations from memory after a brief exposure. While 8-year-olds tended to place the correct number of weights on the correct pegs, showing that they encoded both distance and weight information, 5-year-olds tended to place the correct number of weights on the wrong pegs, suggesting that they were not encoding distance information. Furthermore, 5-year-olds at stage 1 who were trained to encode distance information did in fact benefit from exposure to conflict problems. Thus, only those who are encoding distance information can benefit from feedback that shows the utility of distance information. The relevance of these ideas to the notion of developmental precursors is discussed in chapter 5.

Another candidate transition mechanism is *generalization*, the extension of knowledge to contexts outside of the ones in which the knowledge was acquired. According to Klahr and Wallace (1976), children develop by remembering details about events, detecting regularities in those events, and eliminating redundancies in processing, all of which result in more general knowledge.

A final proposal for a transition mechanism is *representational redescription*. Karmiloff-Smith (1992) argued that much of cognitive development is driven by redescribing existing cognition at higher levels of abstraction. This serves to make implicit knowledge more explicit and potentially available to other parts of the cognitive system.

Integration of transition mechanisms

Because the foregoing list of transition mechanisms appears to be fairly comprehensive, it might be interesting to ask whether these mechanisms could possibly be integrated in some way. Or are they simply independent and competing proposals? Just as it was possible to map Piaget's transition theory onto the cascade-correlation algorithm, I believe that it is possible to integrate the various contemporary proposals into a computational description based on cascade-correlation. Such an exercise

shows how the different proposed mechanisms may be related to each other and clarifies whether they do in fact qualify as causes of developmental transitions.

Learning LTM content, whether it consists of rules or other declarative material, is clearly within the grasp of neural-network learning algorithms. As noted in chapter 2, LTMs are encoded in a network's connection weights. The possibility of implementing acquisition of new strategies in the same fashion is quite straightforward. Indeed, just as rules were considered as epiphenomenal descriptions of processes at the subsymbolic level (in chapter 3), so strategies too can be considered as epiphenomenal descriptions. Symbolically formulated strategies and rules are both more in the heads of symbolically minded psychologists than in the heads of the children they are studying. Such high-level descriptions can provide a kind of summary or abstract characterization of what is going on at the network level, but there is a sense in which the characterization is not what is really going on.

Progressive enhancement of WM capacity might be somewhat more difficult to implement in feed-forward learning algorithms. Indeed, systematic neural-network modeling of WM is only just beginning. In chapter 2, I noted that transient activation patterns across a network implement active memory (AM). Although such patterns might well characterize the transient patterns of conscious cognition, they would not suffice for maintaining slightly longer *memories* for these events. I also noted in chapter 2 that recurrent connections in feed-forward networks implement a kind of WM for network processing over time. Still other neural techniques employ intermediate WM units that remain active for a longer period than conventional units, for example, until receiving another input signal (Guigon & Burnod, 1995). Although I am unaware of any efforts to simulate differences in WM capacity in such models, it would seem feasible to do so, perhaps by varying the number of recurrent connections or the number of WM units.

Similarly, it should be possible to simulate the effects of LTM, in terms of both chunking and automatization, within neural-network models. In general, the more of a task that is accomplished in LTM via connection weights, the less there is for other, for example WM, components to undertake.

At first glance, processing speed would seem to be impenetrable by feed-forward neural-network models, because they propagate activation from inputs to outputs in a single time step. However, as noted in chapter 3, auto-associator networks (among other techniques) can be used to implement latencies for responses that are generated by feed-forward networks. Generally, those responses that generate the most error in a feed-forward network require the longest cleanup times in an appended auto-associator network. So it is entirely reasonable to expect that as error decreases logarithmically over training, response latencies would also decrease logarithmically, as they do in developing humans. Endogenous changes in processing speed having more to do with brain maturation could perhaps be implemented by variation in the learning rate in feed-forward networks or in the update rate in auto-associator networks.

Encoding is very straightforward to deal with in neural-network terms. As a feed-forward network is trained, it learns to encode stimulus patterns onto its hidden units and decode the hidden-unit representations onto output units. Thus, encoding, rather than being a cause of LTM change, is more properly viewed as yet another symptom of network learning. A network learns which inputs it should focus on to solve its current problem by reducing error vis-à-vis target-signal feedback.

In a similar fashion, generalization should be regarded, not as an independent cause of developmental transitions, but rather as a natural outcome of network learning. Networks inevitably try to assimilate new patterns to their existing topology and knowledge, that is, generalize, but their accuracy in doing so usually increases with training on representative patterns.

Many of these integrations apply to a wide variety of neural-network algorithms. However, a few transition ideas, such as reflective abstraction and representational redescription, would appear to be uniquely implemented in generative algorithms such as cascade-correlation. In cascade-correlation, newly recruited hidden units receive input from network input units and from any previously installed hidden units. The hidden units thus effectively redescribe developmentally earlier computations (Shultz, 1994). Because high-level hidden units receive both raw descriptions of inputs (through direct cross-connections) and interpreted descriptions from previous hidden units, they permit ever more

sophisticated interpretations of problems in the domain being learned. Cascaded hidden units thus afford the construction of increasingly powerful knowledge representations that were not available to developmentally earlier instantiations of the network.

This integration shows how all of these proposed transition mechanisms can be viewed, not so much as separate causal mechanisms, but as natural byproducts of neural-network functioning. It also shows how they might work together to generate cognitive development.

Rule Learning

To undertake a comparative analysis of connectionist versus rule-based developmental modeling, we need a symbolic rule-learning program. Of the three most prominent of such programs in the academic marketplace today (Soar, ACT-R, and C4.5), I decided on C4.5, a symbolic learning algorithm that builds a decision tree to classify examples (J. R. Quinlan, 1993). The decision tree, in turn, can be easily transformed into production rules. We saw a glimpse of C4.5 in action at the end of chapter 2, where I perversely used it to learn rules for selecting particular neural-network models.

There were four main reasons for selecting C4.5 over other worthy candidates. First, C4.5 has more successful developmental models to its credit than any other symbolic algorithm. There are four such models in the literature, covering the balance scale (Schmidt & Ling, 1996), past-tense morphology in English (Ling & Marinov, 1993; Ling, 1994), grammar learning (Ling & Marinov, 1994), and reading (Ling & Wang, 1996). There is also a simulation of nonconscious acquisition of rules for visual scanning of a matrix (Ling & Marinov, 1994), which is not particularly developmental, and a large number of applications to real-world problems in machine learning and decision support. Second, in the case of two alternative symbolic rule-learning algorithms applied to the same problem, the balance scale, C4.5 produced a better model (Schmidt & Ling, 1996) than did Soar (Newell, 1990). The C4.5 model was superior in the sense that it covered acquisition of all four balance-scale stages, whereas Soar reached only stage 3, and C4.5 provided coverage of the torque-difference effect, which Soar did not, and presumably could

not, cover. In the field of machine learning, this kind of direct, head-to-head competition is often called a *bakeoff*. Third, C4.5 can actually learn rules from examples, just as connectionist models do. It does not need the extensive background knowledge that Soar and ACT-R seem to require to learn new rules. Finally, C4.5 shares other interesting similarities with cascade-correlation. Both algorithms use supervised learning, focus on the largest current source of error, gradually construct a solution, and aim for the smallest possible solution. C4.5 is, in short, the most plausible symbolic rule-learner to choose for a bakeoff with cascade-correlation. It is not in any sense a "straw man" algorithm selected only to showcase the abilities of cascade-correlation. Nonetheless, it would be interesting to see other researchers try alternate rule-based models in the domains used in the present bakeoff.

The C4.5 algorithm is a direct descendant of the ID3 (Induction of Decision trees) algorithm (J. R. Quinlan, 1986). ID3, in turn, was derived from the CLS (Conceptual Learning Systems) algorithm (Hunt, Marin & Stone, 1966).

As we saw in chapter 2, C4.5 processes a set of examples in attribute-value format and learns how to classify them into discrete categories in supervised learning that uses information on the correct class of each example (J. R. Quinlan, 1993). The learned class description is a logical expression containing statements about the values of attributes, and is equally well formulated as a decision tree or as a set of production rules. A decision tree is either a leaf, indicating a class, or a decision node, which specifies a test of a single attribute with a subtree for each value or possible outcome of the test. Unlike related statistical algorithms, such as Classification and Regression Trees (CARTs) (Breiman, Friedman, Olshen & Stone, 1984), C4.5 can form more than two branches at each decision node, at least with discrete attributes. In my experience, this makes for smaller and more sensible trees than is possible with mere binary branching. C4.5 handles continuous-valued attributes as well, but only with binary branching. The basics of the C4.5 algorithm are quite simple, and the key procedure is exceptionally elegant. It is called *learn* in my version, and it has *examples* and *attributes* as arguments:

1. If every example has the same predicted attribute value, return it as a leaf node.

Table 4.2
Hypothetical examples for deciding whether or not to play

Example (day)	Outlook	Tempera-ture	Humidity	Wind	Play?
1	Sunny	75	70	Strong	Yes
2	Sunny	80	90	Strong	No
3	Sunny	85	85	Weak	No
4	Sunny	72	95	Weak	No
5	Sunny	69	70	Weak	Yes
6	Overcast	72	90	Strong	Yes
7	Overcast	83	78	Weak	Yes
8	Overcast	64	65	Strong	Yes
9	Overcast	81	75	Weak	Yes
10	Rain	71	80	Strong	No
11	Rain	65	70	Strong	No
12	Rain	75	80	Weak	Yes
13	Rain	68	80	Weak	Yes
14	Rain	70	96	Weak	Yes

Source: Quinlan, 1993

2. If there are no attributes, return the most common attribute value.

3. Otherwise, pick the best attribute, partition the examples by values, and recursively learn to grow subtrees below this node after removing the best attribute.

Consider the sample classification problem in table 4.2 (from J. R. Quinlan, 1993). It has 14 examples, each characterized by five attributes, one of which is the classification to be learned, in this case whether or not to play outside. Among the predictive attributes, two are discrete (outlook and wind), and two are continuous (temperature and humidity).

The basic idea in C4.5 learning is to find a small tree that reveals the structure of the problem and has sufficient predictive power to generalize to new examples. As with neural networks, small solutions are most likely to avoid overfitting the training data and to provide the best generalization to test patterns. Like most inductive problems, discovering such a tree is not trivial. For example, $4 \times 10^6 = 4$ million trees are consistent with an entirely discrete version of the play example. In the jargon of computer science, such difficult problems are called

NP-complete (nonpolynomial complete) (Hyafil & Rivest, 1976). Generalization is a wonderful thing, but it is by no means clear, even in toy examples like this one, which is the best generalization and how it can be learned in a reasonable time. There is certainly not enough time to generate all possible trees and then choose the smallest or the one that generalizes best.

Thus, it is important in algorithms like C4.5 to make good decisions about which attribute to use to expand each node of the developing tree. Most decision-tree algorithms are greedy, meaning that they do not backtrack. Once an attribute to test has been selected, the tree is stuck with that choice, which underscores the importance of making good choices of attributes to test. Normally, the only information available for choosing a test attribute is the distribution of classes (*play*/*don't play*) in the examples and their subsets. C4.5 looks over those distributions at each node to be expanded and chooses the attribute that provides the most information about classification. The information contained in a message depends on its probability and is measured in bits as minus the base 2 log of that probability. For example, if there are eight equally probable messages about the classification of an example, the information in any one of them is $-\log_2(1/8)$ or 3 bits of information. C4.5 picks an attribute that maximizes the information gained by partitioning the examples on that attribute.

The information gained by a particular partition of example set S is defined as the difference between the current information and the partitioned information:

$$IG(S) = I(S) - IP(S) \tag{4.1}$$

Here the current information is

$$I(S) = -\sum_j P_j \times \log_2 P_j \tag{4.2}$$

and the partitioned information is

$$IP(S) = \sum_i P_i \times I(S)_i, \tag{4.3}$$

with P_j as the proportion of examples in class j and P_i as the proportion of examples in subset i in the total set S.

J. R. Quinlan (1993) reports better results on some problems when information gain is scaled by split information to create a gain ratio:

$$GR(S) = \frac{IG(S)}{SI(S)} \tag{4.4}$$

Split information is the information generated by partitioning the set of examples S into the same number of outcomes o that would be achieved by applying a particular test:

$$SI(S) = -\sum_{i}^{o} P_i \times \log_2 P_i \tag{4.5}$$

A trace of C4.5's learning the examples in the play problem with the gain-ratio option is shown in table 4.3. To provide this trace, I asked for the gain ratios of each attribute to be printed along with the name of the selected attribute and for the resulting partition of examples. In the case of continuous attributes like *temperature* and *humidity*, C4.5 tried a binary split between each consecutive value in the training examples. In each case, it selected the attribute with the highest information gain ratio to partition the examples. As it happens, the first attribute chosen is *outlook*, which partitions the examples into three groups, one of which (*overcast*) has all its examples in one class (*play*). This is the sort of result that C4.5 is always trying to achieve, to create homogeneous classes of examples through its partitions. The attributes of *wind* and *humidity*, respectively, supply the highest gain ratios in the next two rounds. In the case of humidity, which is a continuous-valued attribute, the best split is between 78% and 79%.

The decision tree created by this learning is shown in table 4.4. Reading from left to right and top to bottom, the tree shows the selected attribute, its values, and then further selected attributes and their values, leading eventually to the classes of the predicted attribute. For example, *if the outlook is for rain and the wind is strong, then we won't play*. Production rules for a tree can be created in just this fashion, by following every path from the root of a tree to a leaf, representing a particular class. Each such path creates one rule. Thus, C4.5 learned to classify the play examples correctly with just 5 rules. In the case of this particular tree, all of the training examples are correctly classified, but in general there is no guarantee that this will happen on every learning problem.

Table 4.3
Trace of play problem in C4.5

```
gain of OUTLOOK = 0.156
gain of TEMPERATURE = 0.017
gain of TEMPERATURE = 0.001
gain of TEMPERATURE = 0.048
gain of TEMPERATURE = 0.048
gain of TEMPERATURE = 0.001
gain of TEMPERATURE = 0.001
gain of TEMPERATURE = 0.029
gain of TEMPERATURE = 0.001
gain of TEMPERATURE = 0.017
gain of HUMIDITY = 0.017
gain of HUMIDITY = 0.048
gain of HUMIDITY = 0.092
gain of HUMIDITY = 0.109
gain of HUMIDITY = 0.029
gain of HUMIDITY = 0.017
gain of WIND = 0.049
choose attribute OUTLOOK
partition ((RAIN DAY14 DAY13 DAY12 DAY11 DAY10)
          (OVERCAST DAY9 DAY8 DAY7 DAY6)
          (SUNNY DAY5 DAY4 DAY3 DAY2 DAY1))
gain of TEMPERATURE = 0.650
gain of TEMPERATURE = 0.650
gain of WIND = 1.000
choose attribute WIND
partition ((STRONG DAY10 DAY11) (WEAK DAY12 DAY13 DAY14))
gain of TEMPERATURE = 0.650
gain of TEMPERATURE = 0.797
gain of HUMIDITY = 1.000
gain of HUMIDITY = 0.797
gain of WIND = 0.650
choose attribute HUMIDITY
partition ((79 DAY2 DAY3 DAY4) (78 DAY1 DAY5))
```

Table 4.4
Decision tree learned in play problem by C4.5

```
OUTLOOK
= RAIN
   WIND
   = STRONG ⇒ NO
   = WEAK ⇒ YES
= OVERCAST ⇒ YES
= SUNNY
   HUMIDITY
   = 79 ⇒ NO
   = 78 ⇒ YES
```

One of the options in my version of C4.5 is to use randomly selected attributes and partitions. This allows us to see how effectively the information-optimization technique in C4.5 is working. Over 20 runs on the play problem, the mean number of rules learned with random partitioning, indexed by the number of leaves in the decision tree, was 9.35. The mean proportion of correctly classified training examples was 0.93, as 9 of the 20 solutions produced two or more mistaken classifications. In contrast, with information optimization turned on, we achieved errorless performance and did so with only 5 rules. This shows that the information-optimizing technique in C4.5 is quite successful in producing compact rule sets and correct learning.

One of the parameters in C4.5 is m, the minimum number of examples to be classified under an attribute value. That is, to be used as a new decision node, an attribute must have at least two values, each of which classifies at least m examples. According to J. R. Quinlan (1993), the m parameter was designed to avoid selecting attribute tests in which nearly all of the examples have the same outcome, because that can lead to trees with little or no predictive power. More interesting for our purposes is that developmental modelers use m to control the depth of decision trees. Small values of m create deeper trees, whereas larger values of m create shallower trees. As we will see, however, the use of m in developmental models has not always been consistent across C4.5 models.

With this powerful rule-learning algorithm in our arsenal of simulation weapons, we can now hold a bakeoff pitting cascade-correlation

against C4.5, and occasionally other algorithms used by other researchers when they are available in the literature. In each domain, our interest is in determining whether a connectionist or symbolic algorithm provides the better model of developmental transitions. The three domains that I consider here are the balance scale, conservation, and habituation of attention.

Balance-Scale Stages

The balance-scale problem was described in chapter 3, where I noted that it has become a major benchmark for computational modeling of development. The clarity and replicability of balance-scale phenomena, coupled with the classical developmental appeal of its stagelike character, have led to both rule-based models (Klahr & Siegler, 1978; Langley, 1987; Newell, 1990; Schmidt & Ling, 1996) and connectionist models (McClelland, 1989; Shultz, Mareschal, & Schmidt, 1994). Whereas in chapter 3 we focused on representational issues and the torque-difference effect, here we focus on issues of transition, namely the ability of computational models to capture transitions between the various balance-scale stages. In other words, does a model produce unaided transitions to all four balance-scale stages?

Simulating transitions is not all that easy, and this allowed us to quickly eliminate several of the models that did not quite work. One rule-based model that used hand-written rules to represent each stage did not develop at all (Klahr & Siegler, 1978). Another rule-based model, using a discrimination-learning technique to learn from initial hand-written rules, developed only stage 3 and lacked the other three stages (Langley, 1987). Another, using the Soar program, which learns production rules by chunking together the results of look-ahead search, captured the first three stages, but failed to reach stage 4 (Newell, 1990). A static neural-network model, using back-propagation learning, likewise captured the first three stages, but never permanently settled into the final stage, and instead perpetually alternated between stages 3 and 4 (McClelland, 1989).[3]

Indeed, some researchers attempted to turn an apparent bug into a feature by arguing that, because many people never reach stage 4 of

the balance scale either, these models that failed to reach stage 4 were actually realistic. Unfortunately, this ignores the fact that some lucky (or at least skilled) individuals actually do reach stage 4, which makes it incumbent on any truly comprehensive model to capture that final transition.

The only two models that capture all of the stage transitions on the balance-scale task are a C4.5 model (Schmidt & Ling, 1996) and a cascade-correlation model (Shultz, Mareschal & Schmidt, 1994). Let's see how they do it.

A cascade-correlation model

Our cascade-correlation model was largely inspired by McClelland's (1989) pioneering back-propagation model, but with some critical differences. There were four input units, one bias unit, and two output units in the initial networks (Shultz, Mareschal & Schmidt, 1994). The input units coded information on the number of weights and the distance from the fulcrum at which they were placed on each side of the beam. Of the four input units, one coded left-side weight, a second left-side distance, a third right-side weight, and a fourth right-side distance. Integers from 1 to 5 coded these values. On the output side, there were two units with sigmoid activation functions that represented balance-scale results in a distributed fashion. A left-side-down outcome was coded by excitation of the first output unit and inhibition of the second output unit. A right-side-down outcome was coded by the reverse pattern. A balanced outcome was coded by neutral values on both output units. Our networks typically recruited between one and three hidden units, which also had sigmoid activation functions.

There were 100 initial training patterns, randomly selected from 625 possible five-peg, five-weight problems. Critically, training patterns had a substantial bias in favor of equal-distance problems (i.e., *balance* and *weight* problems, as seen in figure 3.5). On each epoch in output phases, another training pattern was randomly selected and added to the training patterns, subject to the same equal-distance bias. Thus, the training set was gradually expanded, with one new pattern added in each output-phase epoch. This expansion of the training set assumes that the child's learning environment gradually changes and that these changes are

characterized by exposure to more aspects of the balance-scale world. The large bias in favor of equal-distance problems reflects the assumption, originally made by McClelland (1989), that although children have lots of experience lifting differing numbers of objects, they have relatively little experience placing objects at varying discrete distances from a fulcrum. Without this training bias, networks would skip stages 1 and 2 and move directly to stage 3.

Twenty-four randomly selected test patterns were balanced for both problem type and torque difference, so that there were four patterns from each of Siegler's six problem types (as portrayed in figure 3.5). For each problem type, one pattern was selected from each of four different levels of torque difference. At each output-phase epoch, networks were tested with these 24 test patterns. Any test problem in which both outputs were within score-threshold of their correct targets was scored as correct; any other test problems were scored as incorrect. This was the first, and possibly only, time in which torque differences and problem types were unconfounded, thus making stage diagnosis more definitive. We wrote software to diagnose stages by examining the patterns of correctly and incorrectly answered problems, following Siegler's (1976, 1981) rule-assessment method with children.

Stage-diagnosis results revealed that 11 of the 16 networks progressed through all four stages in the correct (1, 2, 3, 4) order. Two other networks progressed through the first three stages, but did not reach stage 4 (1, 2, 3). One network missed stage 3, but got the other three stages in the correct order (1, 2, 4). Another network showed stages (1, 2, 4) with regression back to stage 3 and then to stage 2. And finally, one network showed stage 1 and then stage 2. With continued training beyond our limit of 300 epochs, such networks do tend to converge on stage 4.

Overlap between diagnoses of adjacent stages near transition points reflected the tentative nature of some stage transitions. Most often, there was a brief period of going back and forth between two consecutive stages before a network settled into the more advanced stage.

In summary, these cascade-correlation networks learned to perform on balance-scale problems as if they were following rules. We sometimes observed developmental regressions and stage skipping, and stage transitions tended to be somewhat soft and tentative. Longitudinal studies in

other psychological domains suggest that such phenomena are characteristic of cognitive development (Siegler & Jenkins, 1989). The cross-sectional research designs used with children on the balance scale are not well suited for investigating issues of stage skipping and regression. Stage skipping, in particular, would require very small time slices to verify that children actually missed a stage. Some regression to earlier balance-scale stages has been noted in existing cross-sectional research (Chletsos, De Lisi, Turner & McGillicuddy–De Lisi, 1989; Siegler, 1981).

Unlike McClelland's (1989) back-propagation network, these cascade-correlation networks did not require hand-designed hidden units, segregated with separate channels for weight and distance information. Also in contrast to McClelland's network, cascade-correlation networks could stay in stage 4 without sacrificing earlier progression through stages 1 and 2. As noted in chapter 3, neural-network models, whether static or generative, naturally produce the torque-difference effect.

A C4.5 model

The C4.5 model also employed a five-peg, five-weight version of the balance scale (Schmidt & Ling, 1996). When the predictor attributes were raw integer values of weights and distances, as in the cascade-correlation model, C4.5 was not able to capture the stages seen in children. These four values had to be supplemented with the following three predictor attributes: whether the problem presented an equal number of weights at equal distances from the fulcrum (yes or no), the side with greater weight (left, right, or neither), and the side with greater distance (left, right, or neither). These three additional attributes essentially represent further processing of the raw weight and distance numbers, computed not by the C4.5 algorithm but by the researchers, who happen to know what is important in learning how to make accurate balance-scale predictions. In the training patterns, there was no bias in favor of equal-distance problems, but there was a bias in favor of simple balance problems (with equal weight and equal distance). Because there are only 25 simple balance problems in the total set of 625 problems, these 25 had to be tripled in frequency. The only justification provided for this, and for the explicit coding of simple balance, was the argument that balance is salient to children. The model was run 100 times, starting with an m of 80 and decreasing by 1 on each run until m was equal to 1 for the last 20

runs. As expected, the progressive decrease in m created deeper and deeper trees until completely correct classification was achieved when m became 1. The authors considered decreasing m to implement an increase in some unspecified mental capacity.

Rule diagnosis was carried out as in the cascade-correlation simulations and was found to reproduce the correct sequence of stages. Because rule diagnosis depends somewhat on the order in which the stage criteria are applied, two different orders were tried, one with higher stages having priority over lower stages and another with the reverse set of priorities. With the former order (4, 3, 2, 1), there was no stage skipping and no regression; with the latter order (1, 2, 3, 4), there was some regression from stage 3 to stage 2, but no stage skipping. A torque-difference effect was found only at stage 3, but not at the other three stages. Rules at each stage were found to be similar to those formulated by Siegler (1976) from his experiments with children.

To simulate the torque-difference effect, the predictor attributes of which side had greater weight or greater distance were converted to continuous variables by subtracting the right-side value from the left-side value. Under these conditions, a torque-difference effect was found at every one of the four stages. A sample decision tree that generates stage 3 performance is presented in table 4.5. At stage 3, children emphasize weight and distance information about equally, but succeed only on

Table 4.5
Decision tree learned on the balance-scale problem by C4.5 at $m = 50$

EQUAL WEIGHTS AND EQUAL DISTANCES
$=$ YES $\Rightarrow$ BALANCE
$=$ NO
  GREATER WEIGHT
  ≤ -1
    GREATER DISTANCE
    $\leq 1 \Rightarrow$ RIGHT SIDE DOWN
    $> 1 \Rightarrow$ LEFT SIDE DOWN
  > -1
    GREATER DISTANCE
    $\leq -1 \Rightarrow$ RIGHT SIDE DOWN
    $> -1 \Rightarrow$ LEFT SIDE DOWN

Adapted from Schmidt and Ling, 1996

simple problems in which weight and distance information do not conflict. An English gloss of one of the rules from the decision tree in table 4.5 is as follows: *if there are not equal weights and distances on each side, the right side has one or more weights than the left side, and the left-side distance is one or less than the right-side distance, then predict that the right side should go down*. Even though such rule sets cover the torque-difference effect, it is apparent that they no longer resemble the rules formulated for children, emphasizing, as they do, weight and distance differences between one side and the other.

On the positive side, this C4.5 model does cover the basic psychological phenomena in the balance-scale literature—the stage transitions and the torque-difference effect—and it is the first and only symbolic rule-based model to do so. On the negative side, the reasons for C4.5 coverage do not seem as natural or principled as those behind the coverage achieved by cascade-correlation networks. First, to capture the stage transitions, it is necessary to extensively preprocess predictor attribute values, with codes for equal weights and distances and explicit comparisons of one side to the other on both weight and distance. Second, to ensure that weight information is initially given more attention than distance information, it is necessary to list the weight attributes before the distance attributes, thus capitalizing on the arbitrary characteristic of C4.5 to break ties in information gain by picking the first-listed attribute. Third, to capture the torque-difference effect, it is necessary to use continuous-valued weight and distance differences among the predicting attributes. This has the unfortunate side effect of rendering the rules learned by C4.5 unrealistic in comparison to those diagnosed in children. Fourth, unlike cascade-correlation models, there is no variation in performance. Every run at a given level of m produces exactly the same decision tree. Finally, developmental transitions depend entirely on decreasing the m parameter to create ever deeper decision trees. It is currently unknown what sort of mental capacity m corresponds to, but worse yet, the m parameter has to be increased, rather than decreased, to cover other developmental phenomena (Ling, 1994; Ling & Marinov, 1993). To date, it has not been explained how and why this unspecified mental capacity increases for some developmental phenomena and decreases for others.

Conservation Acquisition

The conservation problem and its associated psychological phenomena were described in chapter 3, where it was noted that production rules can be written that mimic conservation responses of both younger and older children. Here I describe a bakeoff competition between cascade-correlation and C4.5 to determine their relative success in capturing these phenomena while actually acquiring conservation. Only the cascade-correlation model has been published; the C4.5 model is created here especially for the bakeoff.

As noted in chapter 3, inputs to the cascade-correlation networks included descriptions of how the rows appear in terms of length and density, both before and after one of them is transformed, as well as the nature of the transformation and the identity of the row to which it is applied (Shultz, 1998). Row lengths and densities were indicated by real numbers in the range of 2–6. On the output side, the networks had to learn to predict the identity of the row that had the greater number of items, or whether the two rows had an equal number of items, where number was equal to the product of length and density. Making these problems more difficult, but also more realistic, was that the initial rows could be either equal or unequal in number. Otherwise, merely learning to give a conservation-of-equality answer to every problem becomes really trivial. For each network, 420 training problems and 100 test problems were randomly selected from the 600 possible conservation problems of these sizes.

As might be guessed from the knowledge-representation analyses at the end of chapter 3, these networks did succeed in learning how to conserve. At the end of training, they got virtually all of the training patterns correct and a mean of 95% of the test problems correct, indicating that the successful performance was not merely a matter of memorizing the training patterns. Conservation acquisition for a representative network is presented in figure 4.1 in terms of the proportion correct on training and test problems. A sudden increase in conservation performance is evident after recruiting the second hidden unit, indicated by the second triangle. A regression analysis documented that these networks in general showed a large, sudden jump in performance that

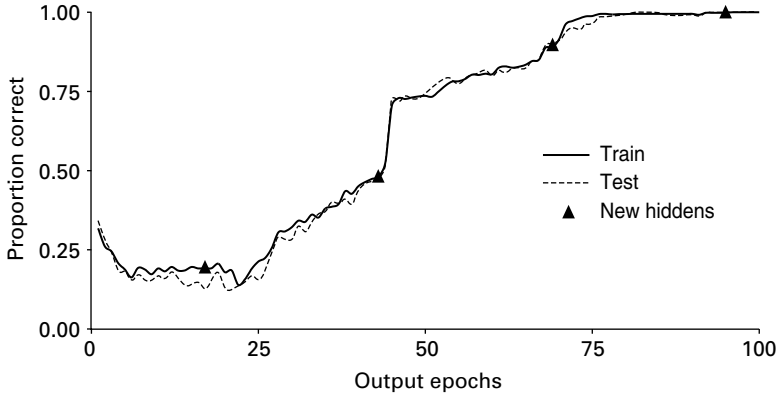

Figure 4.1
Acquisition of conservation in a representative cascade-correlation network.

mirrored that observed in a longitudinal study of children (Raijmakers, van Koten & Molenaar, 1996). Finally, as noted earlier, the cascade-correlation networks were able to cover a variety of other conservation phenomena, including the problem-size effect, length bias, and screening effect.

Because the training patterns used in the cascade-correlation simulations were designed to be comprehensive and neutral with respect to theories and models, the first C4.5 model was also trained with them, with suitable modification in formats. Because training patterns were randomly selected, it was meaningful to perform multiple runs. This first C4.5 model, trained on the same examples as were the cascade-correlation networks, yielded a mean over 20 runs of only 40% correct on training patterns and 35% correct on test patterns.

Now, a failed model is not by itself particularly informative. I sometimes tell my students that building a failing model is about as meaningful as tripping over a rock—anyone can do it, and it doesn't mean much. So my strategy was to change the input coding until C4.5 learning became successful. Then we can evaluate what is required to learn successfully in terms of the psychological coverage it provides. Therefore, I next tried coding the conservation problems in relational terms, much like Schmidt and Ling (1996) did for balance-scale problems. Pre- and post-transformation length and density were each coded according to whether

Table 4.6
Conservation decision tree learned by C4.5 with relational coding

```
LENGTH1
= (L > R)
   LENGTH2
   = (L = R) ⇒ EQUAL
   = (L < R) ⇒ LEFT
   = (L > R) ⇒ LEFT
= (L < R)
   TRANSFORM
   = LEFT
      LENGTH2
      = (L < R) ⇒ RIGHT
      = (L > R) ⇒ RIGHT
      = (L = R) ⇒ EQUAL
   = RIGHT
      LENGTH2
      = (L = R) ⇒ EQUAL
      = (L < R) ⇒ RIGHT
      = (L > R) ⇒ RIGHT
= (L = R)
   LENGTH2
   = (L > R)
      DENSITY2
      = (L = R) ⇒ LEFT
      = (L < R) ⇒ EQUAL
   = (L < R)
      DENSITY2
      = (L = R) ⇒ RIGHT
      = (L > R) ⇒ EQUAL
```

the first row had more than the second, the second had more than the first, or the two rows were the same. This produced C4.5 decision trees with 100% correct responses on both training and test patterns.

A representative decision tree from these runs is presented in table 4.6, where the rows are referred to by *L* for *left* and *R* for *right*. An English gloss of one of the rules in this tree would read as follows: *if the left row is longer than the right row before the transformation and shorter than the right row after the transformation, then the left row has more items*.

All of the C4.5 runs produced rules like this, none of which made any reference to either the transformation that was applied or the identity of transformed row, both of which are supposed to be critical in older children's successful conservation performance (Piaget, 1965). Thus, even though these C4.5-generated rule sets afford completely correct conservation performance, they do not constitute psychologically realistic knowledge representations. In contrast, the knowledge representations acquired by cascade-correlation networks were quite realistic in terms of a shift from an early focus on how the rows looked (their length and, to a lesser extent, their density) to an eventual focus on the identity of the transformed row (left or right) and the nature of the transformation applied to it (refer to tables 3.13 and 3.14).

In summary, the conservation bakeoff between a neural-network model (cascade-correlation) and a symbolic rule-learning model (C4.5) resulted in clear favor of the subsymbolic neural-network model. First, C4.5 could not learn the basic structure of conservation knowledge from raw inputs, just as it could not learn the balance-scale problem from raw inputs. Aided by relational coding of the inputs, C4.5 can learn conservation and generalize effectively, but the rules that it learns are psychologically unrealistic. They do not look at all like the rules diagnosed in children, nor do they cover other psychological phenomena, such as problem-size, length-bias, and screening effects. In contrast, cascade-correlation networks learn and generalize well, even with raw conservation inputs, thus again demonstrating their ability to uncover the essential structure of a problem domain. They also achieve sensible knowledge representations, and cover a variety of conservation phenomena, such as a sudden jump in performance and the problem-size, length-bias, and screening effects.

Habituation of Attention

The third and last case study of transition concerns the habituation of attention in infants. Unlike the cases of conservation and the balance scale, the transition here occurs over the space of a few minutes rather than a few years. In habituation experiments, an infant is repeatedly or continuously exposed to a stimulus until he or she grows tired of it. It is

assumed that the infants build categories for such stimuli and that they will subsequently ignore stimuli corresponding to their categories and concentrate on stimuli that are relatively novel (Cohen, 1973; Sokolov, 1963). Such a mechanism is of obvious adaptive value in promoting further cognitive development. Recovery of attention to novel stimuli is typically called dishabituation. The experimental paradigm is sometimes called familiarization if the exposure to stimuli is insufficient to cause complete habituation.

These processes of habituation and familiarization are typically discussed in terms of recognition memory. If there is substantial recovery of attention to a novel test stimulus, then that stimulus is considered to be novel. But if there is little or no recovery of attention, then the stimulus is considered to be recognized as a member of a familiar category. During the period of habituation or familiarization, there is typically an exponential decrease in infant attention.

Because habituation has made it possible to assess a wide range of perceptual and cognitive abilities in nonverbal, response-impoverished infants, it is one of the most important methodologies in developmental psychology. Because of their performance in habituation experiments, infants have been credited with the ability to perceive form and color, complex patterns, faces, and relations; to learn categories and prototypes; to perceive perceptual constancies; to know about objects and causes; and to build both short- and long-term memories for events (e.g., Cohen, 1979; Haith, 1990; Quinn & Eimas, 1996).

Although neural-network techniques for modeling habituation have been available for some time (Kohonen, 1989), it is only recently that they have been applied to habituation in human infants. Encoder networks, which learn to reproduce their inputs on their output units (see chapter 2), have been shown to simulate habituation and dishabituation effects in infants (Mareschal & French, 2000; Mareschal et al., 2000). In these networks, relations among stimulus features are encoded in hidden-unit representations, and the accuracy of these representations is tested by decoding the hidden-unit representations onto output units. The discrepancy between output and input representations is computed as network error. Familiar stimuli produce less error than novel stimuli, which presumably deserve further processing and learning. Hidden-unit

representations in these encoder networks enable prototype formation, generalization, and pattern completion (Hertz et al., 1991).

To illustrate this work, I focus on one particular data set that has generated at least nine different connectionist simulations in the last two years, in spite of the claim by the original infant researchers that these data were unsuited for a connectionist approach. They claimed instead that their data, concerning infant habituation to sentences in an artificial language, required a rule-and-variable explanation (Marcus, Vijayan, Bandi Rao & Vishton, 1999). It was presumably this challenge that set off the rather large number of connectionist-modeling efforts in a very short time. The connections-versus-symbols nature of the controversy swirling around this data set fits well with the bakeoff theme of the present chapter.

Marcus et al. (1999) familiarized seven-month-old infants to three-word artificial sentences and then tested them on novel sentences that were either consistent or inconsistent with the familiar pattern. The basic design of their three experiments is shown in table 4.7. In experiment 1, infants were familiarized to sentences with either an ABA pattern (e.g., *li ga li*) or an ABB pattern (e.g., *ga ti ti*). There were 16 sentences, constructed by combining four A-category words (*ga*, *li*, *ni*, and *ta*) with four B-category words (*ti*, *na*, *gi*, and *la*). After the infants became familiar with a sentence pattern, they were tested with two sentences containing novel words that were either consistent or inconsistent with the familiar sentence pattern.

When an infant looked at a flashing light to the left or right, a test sentence was played from a speaker placed next to the light. This test sentence was played until the infant either looked away or 15 seconds

Table 4.7
Design of the experiments by Marcus et al. (1999)

	Experiments 1 & 2		Experiment 3	
Sentences	Condition 1	Condition 2	Condition 1	Condition 2
Familiar	ABA	ABB	ABB	AAB
Consistent	ABA	ABB	ABB	AAB
Inconsistent	ABB	ABA	AAB	ABB

elapsed. The basic finding was that infants attended more to inconsistent novel sentences than to consistent novel sentences, showing that they distinguished the two sentence types.

Experiment 2 was the same except that the particular words were chosen more carefully to ensure that phoneme sequences were different in the familiarization and test patterns. Finally, experiment 3 used the same words as did experiment 2, but in contrastive syntactic patterns that each duplicated a consecutive word: AAB or ABB. The purpose of experiment 3 was to rule out the possibility that infants might have used the presence or absence of consecutively duplicated words to distinguish between the two sentence types.

In all three of these experiments, infants attended more to inconsistent than to consistent novel sentences. But what is the best theoretical account of these data? Is the infant cognition underlying these syntactic distinctions based on symbolic rules and variables or on subsymbolic connections?

Marcus et al. (1999) claimed that these grammars could not be learned by what they called the *statistical* methods common to standard neural networks.[4] They also tried some neural-network simulations using SRNs (refer to chapter 2), which proved to be unsuccessful in capturing the data. They argued that only a rule-based model could cover their data. "We propose that a system that could account for our results is one in which infants extract algebra-like rules that represent relationships between placeholders [variables] such as 'the first item *X* is the same as the third item *Y*' " (1999, 79). They hedged a bit by noting that their data might also be accounted for by so-called structured neural networks that implement explicit rules and variables in a neural style: "The problem is not with neural networks per se but with the kinds of neural networks that are currently popular. These networks eschew explicit representations of variables and relations between variables; in contrast, some less widely discussed neural networks with a very different architecture do incorporate such machinery and thus might form the basis for learning mechanisms that could account for our data" (1999, 79–80).

Indeed, one of the nine successful neural-net simulations is of this structured sort (Shastri, 1999). This model had explicit variable binding, implemented by temporal synchrony of activations on units that

represented sequential positions of words and other units that represented arbitrary binary word features. The model had no explicit rules in the sense of symbolic *if-then* propositions. The network learned to represent, for example, an ABA pattern by firing the first-position unit synchronously with the third-position unit. Such a network would seem to generalize very well to any novel sentences of three words, regardless of the particular features of the words used. However, this structured network is built by hand, and the feedback signals that it requires to learn about the position of words in a sentence are psychologically implausible. Although infants certainly hear sentences with words in various positions, there is no systematic feedback about the positions of the words.

A review of the other eight connectionist models of the Marcus et al. data (Shultz & Bale, 2001) reveals that only one of them is structured (Gasser & Colunga, 1999), although perhaps not as extensively as the Shastri model. Seven of them are standard unstructured neural networks of the sort covered in chapter 2: SRNs (Altmann & Dienes, 1999; Christiansen & Curtin, 1999; Negishi, 1999; Seidenberg & Elman, 1999), cascade-correlation networks (Shultz, 1999; Shultz & Bale, 2001), and auto-associator networks (Sirois et al., 2000). All nine of these connectionist models cover the basic findings of the infant data in terms of learning to distinguish between consistent and inconsistent sentences, but many of them postulate assumptions about either the training or the network architecture that may not be agreeable to everyone. At this point though, there is no question that the infant data can be covered by unstructured neural networks. A symbolic rule-based system is most certainly not required and, indeed, has not even been reported as successful. Continuing in our comparative spirit, we now examine a bakeoff competition between my favorite connectionist model of these data and the most successful symbolic rule learner on developmental problems, C4.5.

The connectionist model uses an encoder version of cascade-correlation (Shultz & Bale, 2000, 2001), as described in chapter 2. The network basically learns to recognize the sentences to which it is exposed during a habituation phase. Error on these training sentences decreases exponentially during the habituation phase, mimicking the decrease in attention

seen in many infant-habituation experiments. After training, error on the consistent test sentences is significantly less than that on the inconsistent test sentences, capturing the basic finding in the Marcus et al. (1999) experiments. Moreover, this consistency effect generalized beyond the range of the training patterns. The words and sentences are those used with the infants, with words coded in a realistic fashion by the sonority of the phonemes. Sonority is the quality of vowel similarity, as defined by perceptual salience and openness of the vocal tract. The proportion of network reversals (.0667) of the consistency effect (more error to consistent than to inconsistent test patterns) was eerily close to the proportion of infants that preferred to look at consistent patterns (.0625).

My first C4.5 model treated the Marcus et al. (1999) sentences as a concatenation of symbols, e.g., *li ga li* or *ga ti ti*. The predicting attributes were the three word positions, and the predicted attribute was the artificial grammar, for example, ABA or ABB. Presented only the ABA sentences, C4.5 unsurprisingly produced a decision tree with no branches and one leaf, labeled ABA. Note that it could have done this even with only one or two sentences, and that it does this immediately, as if in a single trial. C4.5 does not require the full complement of 16 sentences, nor does it show any exponential decrease in error. So far, this shows only that C4.5 is unsuited to modeling habituation or any other form of recognition memory; it is really a discrimination learner or classifier.

Consequently, my next effort was to convert the problem into a discrimination (or classification) problem, essentially by including the 16 contrasting ABB sentences in the training set. Before objecting too strenuously to such a major change in the task, please note that some connectionist models also employed similar changes for pretraining their SRNs (Seidenberg & Elman, 1999; Christiansen & Curtin, 1999), not in such a bald-faced way, but still making a shift to a discrimination-learning paradigm. Perhaps someone can construct a convincing argument for why a C4.5 model deserves a discrimination version of this task.

In any case, it is interesting to see how C4.5 does with a discrimination version of the artificial-syntax task. The decision tree that it generates is shown in table 4.8. It focuses only on the third word position and uses the words it sees in that position to distinguish ABA from ABB sentences. This is nothing at all like the first-word-matches-third-word rule

Table 4.8
Decision tree generated by C4.5 on a discrimination version of the syntax task

3
= LA ⇒ ABB
= GI ⇒ ABB
= NA ⇒ ABB
= TI ⇒ ABB
= TA ⇒ ABA
= NI ⇒ ABA
= LI ⇒ ABA
= GA ⇒ ABA

Table 4.9
Decision tree generated by C4.5 on a discrimination version of the syntax task with relational coding

13
= DIFFERENT ⇒ ABB
= SAME ⇒ ABA

envisioned by Marcus et al. (1999), but it is quite ingenious nonetheless in its relentless focus on an obvious difference between ABA and ABB sentences. Needless to say, this solution will not generalize at all well to the novel-word test sentences, as it depends entirely on the words encountered in the third position in the training sentences.

Okay, how about a relational coding scheme like those in the balance-scale and conservation simulations with C4.5? In this case, the important relations are between word positions in the sentences. For ABA sentences, positions 1 and 2 are *different*, 1 and 3 the *same*, and 2 and 3 *different*. Coding an ABB sentence in a similar fashion and running C4.5 yields a decision tree that performs as strongly as Marcus et al. would presumably like, as shown in table 4.9. If words 1 and 3 are the *same*, then you have an ABA sentence; if *different*, then an ABB sentence. This will generalize perfectly, but consider the drawbacks:

• We have provided C4.5 with the solution in our relational coding, coming very close to the rule-writing tendencies of many symbolic-computation adherents.

• No more than a single exposure to as few as two sentences generates perfect knowledge of the problem.

• No reversals of the consistency effect would be possible with this knowledge.

Not quite willing to give up, I also let C4.5 try the sonority-coding scheme used in the cascade-correlation simulation of the syntax problem. Does the gander do as well as the goose? Not really. With this coding scheme, C4.5 produces a tree that is correct on only 62.5% of the training sentences and fails entirely on the test sentences, whether consistent or inconsistent. Moreover, the tree contains rules of the following sort:

If $C1 < -5, C3 < -5,$ and $C2 > -6,$ then syntax is ABA.

If $C1 < -5$ and $C3 > -6,$ then syntax is ABB.

Here C1 refers to consonant 1, C2 to consonant 2, and so on, and the integers refer to sonority values.

To summarize, C4.5, the first reported full-blown symbolic rule-learning system to be applied to a data set that was claimed to be amenable only to symbolic rule-based models, does not fare well. First, C4.5 does not model habituation, because it quickly and trivially learns to generate the only category to which it is exposed. When asked to model a different task, namely discrimination (its specialty), it does not learn the desired rules except when virtually given the rules by the coding scheme. Writing rules to fit psychological data is one thing; creating an automatic rule-learner that fits psychological data is quite a bit more difficult. In contrast, cascade-correlation learns a realistic interpretation of the syntax-habituation problem with a realistic stimulus-coding scheme in a way that captures all the main features of the little that is currently known about this problem: exponential decrease in attention, post-habituation preference for inconsistent patterns, a slight tendency to prefer consistent patterns, and generalization beyond the range of the training patterns.

Conclusions from the Case Studies

Do these three case studies prove that symbolic rule-based approaches cannot handle developmental transitions? No, because we have not

provided a logical proof that rule learning cannot work, nor have we tried all of the available rule learners. We have most certainly not tried those rule learners yet to be designed. The case studies do, however, highlight the sorts of problems that would confront any candidate rule-learning program. Inducing rules is a very complex business, and even the arguably best current rule-learning algorithm for developmental phenomena does not always learn the rules that the modeler would like to see. These problems seem formidable indeed compared to the relative ease and naturalness with which current connectionist models acquire the relevant developmental transitions. With these case studies behind us, I next turn to an examination of four basic theoretical issues about transition.

How Can Anything Genuinely New Be Learned?

The constructivist view of cognitive development holds that children build new cognitive structures by using their current structures to interact with the environment. Such interaction with the environment forces adaptation to environmental constraints, and the adaptation of existing cognitive structures results in new, more powerful cognitive structures. Constructivism was inspired by Kant's resolution of the rationalist-empiricist debate and served as the basis for Piaget's (1977) theory of cognitive development and the considerable body of empirical research that followed.

Fodor (1980) punctured a neat hole in the constructivist balloon by arguing that a constructivist account of cognitive development was not logically coherent. Interestingly, Fodor's argument was based essentially on computational considerations. None of the computationally precise learning algorithms that Fodor was familiar with in the late 1970s were capable of learning anything genuinely new, in the sense that they had to possess the representational power to describe anything that they could learn. For example, to learn the concept of *red square*, a learning system must already be able to represent *red*, *square*, and *conjunction*. Without such representational abilities, the learning system could not build hypotheses such as *red square* to test against the evidence. If these hypothesis-testing algorithms possessed the three representations of *red*,

square, and *conjunction*, they could combine them to form the hypothesis *red square* and then test that hypothesis against the available evidence.

The implication for cognitive development was that children could not construct anything genuinely new through experience-based learning mechanisms. This was meant as a fatal blow to Piaget's constructivist account and prima facie evidence in favor of a more nativist view that children come equipped with the full cognitive powers of an adult. Just as in Chomskyan-inspired psycholinguistics, the argument was essentially that *if it cannot be learned, then it must be innate*. Never mind that a full nativist account was never actually specified.

In practice, Fodor's argument against the possibility of constructivism was largely ignored by many developmental researchers, who continued to work, at least implicitly, within a constructivist framework. However, the fact that Fodor's argument had never been successfully countered provided a disturbing backdrop for much of that research. I refer to it as *Fodor's paradox* because it seems to be a fundamentally sound argument against what seemed to be an inherently correct assumption that cognitive development is driven by experience. Contemporary updates of Fodor's view indicate that it was not a one shot deal (Bloom & Wynn, 1994; Marcus, 1998).

Recently it has been argued that generative networks, such as cascade-correlation, have the capacity to escape from Fodor's paradox (Mareschal & Shultz, 1996; Quartz, 1993). After recruiting new hidden units, these generative networks become capable of representing relationships that they could not possibly represent previously. Indeed, their lack of early representational ability likely produced a stagnation of error reduction and triggered the recruitment process.

In contrast, it would seem that static neural networks fail to escape Fodor's paradox because the range of functions they can learn is limited by their initial network topology (Mareschal & Shultz, 1996; Quartz, 1993). At first glance, it might appear that static networks could also escape Fodor's paradox. For example, the fact that static networks are able to learn new representations might allow them to escape. While it is true and amply demonstrated that static networks can learn new representations (Elman et al., 1996), the computational power of these static

networks is clearly limited by their initial topology. That is, they can learn only those functions that can be represented within that initial topology. Thus, Fodor's (1980) view, that one must be able to represent the hypotheses that can possibly be tested, still applies.

A well-known example in the connectionist literature is that a static network using back-propagation of error cannot learn an exclusive-*or* problem unless the network has at least two hidden units.[5] In terms of the example just discussed, a static network with only one or no hidden units could learn to represent *Red and square* or *Red or square*, but not *Either red or square, but not both red and square*.

Exclusive-*or* is a nonlinear problem in the sense that no linear combination of weights can be learned to solve it. As noted in chapter 2, exclusive-*or* can be considered as a two-bit parity problem, in which the network's output unit should respond positively only if there are an odd number of 1s in the input. Parity problems with more than two bits of input can also be constructed and require even more hidden units to solve because of their increasing nonlinearity with increasing numbers of input bits. In general, the greater the degree of nonlinearity in the problem to be learned, the more hidden units required to learn it.

It might be thought that static network designers can escape Fodor's paradox by fitting the network with very large numbers of hidden units. The number of hidden units a network possesses may be taken as a rough index of its computational power. However, as also noted in chapter 2, networks that are too powerful have a tendency to memorize their training patterns rather than abstract useful generalizations about them, i.e., these oversized networks tend to generalize poorly. Poor generalization is considered fatal for both engineering applications and cognitive modeling because the network fails to deal effectively with novel patterns.

It is worth considering whether evolutionary forces might endow static biological networks with just the right amount of computational power. For some essential skills, such as perception or language or basic cognitive abilities like object permanence, evolutionary pressure might well have done just that. But it is doubtful that evolution could have prepared us for all of the cognitive tasks that we face in rapidly changing environments. The learning of mathematics or computer programming may be cited as convincing examples. It is much more likely that we require

flexible network modules that can grow as needed to adapt to a variety of novel problems. Quartz and Sejnowski (1997) argued that evolution has prepared us for flexible learning. In such cases, it seems important for learning algorithms to find the proper size and topology of a network to facilitate learning.

Another possible argument that static networks can escape Fodor's paradox is to imagine that the process of increasing initially small random weights during learning is really the same as the recruitment process in generative networks such as cascade-correlation. Effectively, a hidden unit that has little influence may, through learning, have its importance dramatically increased. Static networks certainly do change their weights in response to learning pressures. However, the process whereby an initially meaningless unit with random weights is progressively brought on line by learning appropriate weights is a very different process from that of hidden-unit recruitment in generative networks (Shultz & Mareschal, 1997). Even the apparently useless units in a static network (i.e., those with small random or inappropriate weights) are contributing to the total processing of the network. As seen in equations 2.5–2.11, these units send residual activation to other units, and they are contributing to the calculation of the error terms used to adjust other weights in the network. Hence, they are an integral part of the computational power available to solve the problem. In generative networks, however, units not installed in the network do not contribute to the functioning of the network in any way. They are not sending any activation. Nor are they factored into the error-adjustment algorithms. They are simply not part of the network module. In the early stages of network learning, cascade-correlation networks are searching a decidedly smaller weight space than are static networks of the same size as what the cascade-correlation network may eventually achieve. As noted in chapter 2, this ability to start small and increase in size as needed may provide cascade-correlation with an advantage over static networks in learning difficult problems.

Thus, static networks do not share the ability of generative networks to escape from Fodor's paradox. A progressive growth in network computational power appears to be necessary for escaping this paradox.

Because of the demonstrations that generative neural network models can escape Fodor's paradox and model human cognitive development,

we can conjecture that constructivist models of cognitive development are indeed possible. Of course, evidence that they are the best explanation for cognitive development awaits future psychological and computational study.

Because synaptogenesis is so pervasive and important in brain development (Quartz & Sejnowski, 1997), it is critical for neural-network models to be able to grow as well as to learn. Without such growth capabilities, it is doubtful that constructivist cognitive development could occur, as argued by Fodor (1980).

The Relation between Learning and Development

Most of the theories and models discussed in this book and throughout the field of psychological development assume, despite Fodor's paradox, that development occurs through some kind of learning of long-term memory elements. This raises the question of whether we need the term *development* at all in order to fully understand the psychological changes that children go through. Why not just focus on learning per se?

Despite how tempting that idea may seem, many theorists of psychological development persist in using the terms *learning* and *development* as if they were two different things. In many cases, however, they do this without really making a clear distinction between the two. In *the major* effort to understand development in terms of neural networks, for example, Elman et al. (1996) end up explaining development in terms of weight adjustment within static neural networks. They do make a good argument that there is development underlying human cognitive change, but the simulations they report consist of learning models that attempt to capture developmental data. The ability of a neural network to mimic developmental data does not, in itself, make it a developmental model. Static back-propagation networks only implement learning, even when they produce nonlinear changes in performance.

For generative network models, like cascade-correlation models, which grow as well as learn, it is possible to draw a clear distinction between learning and development. Interestingly, this distinction is one that can give computational explicitness to the verbally formulated ideas of a number of developmental theorists.

Learning can be defined as parametric change *within* an existing processing structure in order to adapt to information from the environment (Sirois & Shultz, 2000). This definition is compatible with definitions offered by a wide range of theories, including nativist (e.g., Fodor, 1980), empiricist (White, 1970), and constructivist (e.g., Piaget, 1980). In contrast, development can be defined as a change *of* an existing structure to enable more complex parametric adaptations (Sirois & Shultz, 2000). Thus, development is a qualitative change in the structure supporting cognition, and learning is a quantitative change in parameter values within a particular cognitive structure. Such a distinction is basically compatible with those made in Piaget's (1980) theory of abstraction, Karmiloff-Smith's (1992) theory of representational redescription, Carey's (1985) theory of conceptual change, and Liben's (1987) general discussion of the difference between learning and development.

The big difference is that now we have a clear computational view of what the distinction might mean. Learning occurs via connection-weight adjustment (quantitative parameter change), and development via hidden-unit recruitment (qualitative change in structure). Interestingly, this view implies that, although learning may occur without development, as when a problem can be learned without recruiting additional hidden units, development always involves learning. The reason for the latter claim is that each recruitment of a hidden unit requires learning to find and train the recruited unit and then more learning to determine how to best incorporate this new representation device into the overall solution. The first kind of learning occurs in the input phase of cascade-correlation, and the second in the ensuing output phase. On this view, it is correct to say that a lot of psychological growth results from a combination of learning and development, and that development incorporates learning. It is also correct to say that some phenomena involving psychological change in children may well be due to learning alone. At the present state of the art, decisions about whether a particular psychological change in children is due to learning or to development may be greatly aided by accompanying generative-neural-network models. A model that fits the psychological data can be examined to see if hidden units have been recruited.

How Might Innate Determinants Operate?

Nativist approaches to psychological development have been around for a long time and continue to be influential, particularly in discussions of language acquisition and early infant competence. Contrary to the common view that connectionist approaches are antinativist because of their emphasis on learning, an important connectionist contribution to the study of innate factors in development concerned the different ways in which psychological processes could be construed to be innate (Elman et al., 1996). Elman et al. argued that innateness can occur in representations, architectures, or timing. Their focus was not so much on identifying what is innate, but more on expanding consideration of how things could be innate.

The classical view, shared by nativists, empiricists, and constructivists alike, is that innateness occurs at the level of knowledge representations. The basic idea of representational innateness is that children have domain-specific knowledge that is somehow controlled by a genotype. Recent proposals of this sort, for example, have pointed to innate knowledge of syntax (Pinker, 1994), arithmetic (Wynn, 1992), and physics (Spelke, 1994). The neural-network analog to such representational innateness is to have many or all of a network's connection weights specified before learning starts.[6]

Indeed, some interesting work along these lines has documented interactions between evolution and learning in neural networks that are allowed to reproduce as well as to learn. In a population only those networks that are most fit, in terms of learning whether an input pattern is symmetrical or not (Belew, McInerney & Schraudolph, 1991) or learning to find food (Menczer & Parisi, 1992; Nolfi, Elman & Parisi, 1994) were allowed to reproduce, in some studies sexually and in others asexually. Evolution succeeded in preparing successive generations to be better learners, even when the learning task (predicting food location on the basis of its current location and planned network movement) was different from the fitness task (obtaining food) (Nolfi et al., 1994). Not only did evolution improve learning, by selecting more promising initial weights, but learning also accelerated evolution, by flexibly exploring solutions to the fitness task. Interestingly, these findings represent neither

evolution of innate abilities nor Lamarckian transmission of learned knowledge. Rather, the networks were predisposed by evolution to be good learners by the transmitted initial, unlearned connection weights.

However, Elman et al. (1996) disavow representational innateness, arguing that there is insufficient information in the human genotype for it to be feasible. They astutely point out even the molecular parts of the body cannot be fully specified in the genotype, much less large amounts of psychological representations. Evidence is cited that the human body contains 5×10^{28} bits of molecular information, but the human genotype contains only 10^5 bits of information.[7] The implication is that the human genotype does not have enough information to serve as a blueprint for possible innate aspects of language and cognitive development.

Elman et al. also use evidence for the initial equipotentiality of mammalian cortex to discredit representational innateness. The idea that the genotype contains a detailed blueprint for cortical functioning is difficult to maintain against evidence that cortical neurons can serve a variety of functions, depending on experience. The gist of this evidence can be summarized under the maxims "When in Rome do as the Romans do" and "You are what you eat." Compelling demonstrations come from experiments with small mammals that transplant pieces of fetal cortex from one area to another or redirect thalamic inputs from their usual targets to some other cortical location. In such experiments, the cortex takes on properties of the area that it is now in ("When in Rome . . .") or those of the input it receives ("You are what you eat"). It is as if auditory cortex becomes able to see and visual cortex becomes able to hear. If cortical material does not initially know its eventual job and can be recruited for other jobs, then how could its domain-specific content be innately specified?

In a critique of this argument, Marcus (1998) speculates that some kinds of representations could be innate even though no individual neuron has an initially specified role. Instead, he argues that a cell could carry conditional instructions that specify different functions depending on particular conditions such as location and input. However, this counterargument fails because such conditional rules would require vastly more information than the single-function instruction envisioned by Elman et al. (1996).

Another way for something to be innate is in terms of architectural constraints. Elman et al. (1996) break down architectural constraints into unit, local, and global constraints. At the unit level would be features like firing thresholds, transmitter types, and learning rules. Connectionist analogs of such unit constraints would be activation functions, learning rules, and the parameters of learning rules. Local constraints would be things like number of layers of neurons, connection density, and circuitry. Analogs to these constraints in artificial neural networks would be network topologies, including numbers of layers and units. At the global level, there would be constraints from connections between brain regions. These could be implemented in neural networks via modules that may have different jobs and that feed input to other modules. Currently, these kinds of architectural decisions are typically made by the researchers and implemented by hand, except for generative algorithms that create their own topologies. Some evolutionary simulations have successfully explored genotypes involving network architectures (Miller, Todd & Hegde, 1989), parameter values for learning rate and momentum (Belew et al., 1991), and learning rules (Chalmers, 1990).

The third way for something to be innate is in terms of the timing of events in development. An example cited by Elman et al. concerns spatiotemporal waves of cortical development. The locus of maximum neural plasticity begins in the primary sensory and motor areas, migrates to the secondary association areas, and finally to the frontal areas (e.g., Thatcher, 1992). When developed and connected, these regions act as successively higher-level filters of incoming information, from primary to secondary areas, and on to frontal areas. Such progressive spreading of plasticity could be built into static networks, but it is worth noting that generative algorithms like cascade-correlation implement it naturally by freezing the weights to recruited hidden units and by training and recruiting still more hidden units downstream.

While Elman et al. consider representational innateness to be unlikely, they do consider architectural and timing constraints to be reasonable forms of innateness. Indeed, with static networks, most architectural constraints have to be innately specified.

However, it is worth asking whether the brain would come innately wired with all of the networks in place, with the correct size and con-

nectivity, that will eventually be needed for mastering a lifetime of tasks and problems. In order to produce appropriate static network topologies, the brain would seem to require some a priori representation of the problems it will have to learn (Sirois & Shultz, 1999). This is because, as noted in chapter 2, network topology determines the range of functions that can be learned. Thus, innately designed networks, even though having random weights, still imply representational innateness, if only a relaxed version. As Quartz noted, static networks "have built into their architecture a highly restricted hypothesis space that contains the target function, or at least an acceptable approximation to it" (1993, p. 233). Failure of the evolved brain to specify networks of the right topology for a lifetime of learning would run the risk of having to learn many problems with networks that were either too weak, leading to learning failure, or too powerful, leading to generalization failure (see chapter 2). A formal analysis of the brain's ability to specify topology suggests that the probability is virtually nil that the brain can anticipate the right size networks for a realistically wide range of learning problems, many of which were not present in the environment when the brain evolved (Sirois & Shultz, 1999).

In summary, by considering architecture and timing, as well as representations, Elman et al. (1996) have used neural-network research to significantly and creatively enlarge our conception of how psychological development might be innately constrained. Indeed, critics of their contribution (e.g., Marcus, 1998) have missed the main point by attacking Elman et al.'s (1996) critique of representational innateness. Elman et al. have given us new ways and methods to investigate innate determinants of psychological development. However, by implementing only static networks that must be innately designed even though not innately trained, Elman et al. have inadvertently opened the door to a relaxed form of representational innateness. Rather than having the burden of anticipating all learning problems that an organism will face over a lifetime, it would seen preferable to have a brain flexible enough to design its own networks through neurogenesis and synaptogenesis. Quartz and Sejnowski's (1997) recent review of brain development concluded that plasticity is most often found in species that are phylogenetically recent and proximal to humans. They characterize human evolution as moving

towards maximal plasticity rather than towards hyperspecialization. This kind of flexible network construction is better approximated by generative networks, such as cascade-correlation, than by static, innately provided networks.

Generative versus Static Networks

Given the foregoing considerations, is it ever appropriate to use static networks in modeling psychological development? Even if generative networks are more powerful and flexible general learning systems than static networks, there still may be domains in which it is more appropriate to use static networks than generative networks to model development. There are several domains in which generative networks produce better simulations than do static networks (e.g., the balance scale and integration of velocity, time, and distance cues; see chapter 5 for the latter). So far, there have been no demonstrations of the opposite trend, but it is possible that static networks might be superior on some problems. And there may be many domains in which phenomena could be equally well modeled by static or generative networks.

It has been suggested that static networks should be used to model domains that are constant across all individuals and for which evolutionary pressures may have prepared networks with either topology alone or both weights and topology (Shultz & Mareschal, 1997). Examples might include some basic abilities in areas such as vision, audition, spatial and temporal reasoning, causal inference, memory, categorization, and aspects of language. These abilities begin to develop very early in infancy and are found in all cultures. No matter where an infant is born, she will need to develop this knowledge in a form that is consistent across all cultures.

Static networks have been used to model a number of basic infant abilities such as categorization (Mareschal & French, 2000; Mareschal et al., 2000) and object permanence (Mareschal, Plunkett & Harris, 1999; Munakata, 1998; Munakata, McClelland, Johnson & Siegler, 1997). These are abilities found in every infant and might well serve as building blocks for learning more complex tasks.

Cognition that apparently builds on this initial learning tends to vary greatly over the planet. Evolution could not possibly anticipate what every child might eventually need to learn. The learning required of children in a hunting-and-gathering culture is quite different from that required of children learning to program computers. Despite some flexibility in the possible initial network topologies that can be used to learn a task, getting the topology wrong can determine whether a task can be learned or not and how easily and how well it is learned (Mareschal & Shultz, 1996; Quartz, 1993). Thus, the ability to learn a wide range of tasks requires the ability to construct appropriate networks.

In summary, although it is presently difficult to predict what sort of model is likely to produce a better model of a particular domain, it might be that static networks are better for those problems that evolution could correctly anticipate. Generative networks might be preferred for learning problems whose features are less predictable.

General Conclusions about Transitions

A generative connectionist algorithm (cascade-correlation) was found capable of integrating both Piagetian and contemporary proposals for explaining developmental transitions, perhaps the most persistently difficult theoretical problem in the field of developmental psychology. A systematic comparison of this algorithm to the leading rule-based learner (C4.5) over three case studies of development demonstrated a consistent superiority of the connectionist approach. Cascade-correlation learned the essentials in each domain, naturally captured a variety of associated phenomena, and produced knowledge representations that were psychologically realistic. In contrast, C4.5 had difficulty learning unless the examples were coded in a very helpful format, failed to cover associated phenomena, and typically produced knowledge representations that were not psychologically realistic.

Cascade-correlation simulations further showed how it was computationally feasible to implement a constructivist account of development, thus escaping Fodor's paradox about experience-based learning. Using cascade-correlation ideas, I formulated a clear distinction between

learning and development by noting that learning involves quantitative parameter changes within an existing cognitive structure, whereas development involves qualitative changes in the cognitive structures themselves. A recent connectionist account of innate determinants (Elman et al., 1996) usefully enlarged the possibilities to include timing and architectural constraints, while dismissing the more conventional representational innateness. However, without a generative approach, connectionism may inadvertently allow a relaxed form of representational innateness. Finally, there could well be room for both static and constructivist neural-network models of development, with static models being more appropriate for universal knowledge and constructivist models being required for culturally specific knowledge.