Anuario de Psicología
1996, nº 69, 93-113

Constructivism, epistemology and information processing*

David Leiser
Ben-Gurion University of the Negev

The author analyzes the main models of artificial intelligence which deal with the transition from one stage to another, a central problem in development. He describes the contributions of rule-based systems and connectionist systems to an explanation of this transition. He considers that Artificial Intelligence models, in spite of their limitations, establish fruitful points of contact with the constructivist position.
Key words: *Constructivism, Artificial Intelligence, Rule-based systems, Connectionism.*

El autor analiza los principales modelos de inteligencia artificial que dan cuenta del paso de la transición de un estadio a otro, problema central del desarrollo. Describe y señala las aportaciones de los sistemas basados en reglas así como de los sistemas conexionistas para explicar dicha transición. Considera que los modelos de inteligencia artificial, a pesar de sus limitaciones, permiten establecer puntos de contacto muy fructíferos con la posición constructivista.
Palabras clave: *Constructivismo, inteligencia artificial, sistemas basadsos en reglas, conexionismo.*

Over the past several decades, enormous progress has been made in the domain of computer acquisition of new knowledge. The extent of this progress risks to make one loose sight of the true magnitude of the task (see Leiser, 1995), but there is no question that our grasp of knowledge construction is being enriched by the concepts developed in the various branches of Machine Learning and Neural Networks. It is of course impossible to summarize here the vast

* The support of the Center for Health, Safety and Human Factors, Ben-Gurion University is gratefully acknowledged.
Dirección del autor: David Leiser. Department of Behavioral Sciences, Ben-Gurion University of the Negev, Beer-Sheva, Israel 93661. e-mail: dleiser@bgumail.bgu.ac.il.

amount of work pouring forth from those disciplines (see e.g., Carbonell, 1990, Dresher, 1991), in the course of a brief survey. I will limit myself to presenting some important ideas, especially those may be less familiar to the average reader in psychology.

An account of development supposes two components, that may be more or less closely linked: a good description of the lower and higher stages of development, and an account of the transition. Until about fifteen years ago, while psychologists had collected good descriptions of key stages of development, it was not at all clear how to describe the actual passage from one to another. One could not say much about intermediate stages, beyond unhelpfully indicating that they share some properties of both the lower and the higher stage. Extensive progress in the representation of intermediate levels was made since. This progress was mainly carried by two currents. The one is modular, «localist». It analyzes a complete and complex skill as the interplay of a large number of rules that jointly generate the required behavior. The emergence of a better adapted behavior may then be described as the gradual assembling and refining of a set of elementary rules, each one of which may, up to a point, be understood in isolation (Anzai and Simon, 1979; Young, 1976). Progress was also made in the study of transition mechanisms, of which a variety have been implemented (e.g., Holland, Holyoak, Nisbett and Thagard, 1986), though it remains unclear how well they can really account for the range of cognitive structures believed to exist (Leiser, 1990). The second current is formed by the varieties of connectionist models (PDP, auto-associators, self-organizing maps, etc.). These distributed systems have already provided convincing examples of the acquisition of a very diverse array of knowledge. In the following pages, we will try to provide some introduction to both, discussing as we go the epistemological relevance of these mechanisms.

I. Rules-based systems

Skillful behavior consists in doing a variety of actions under the correct conditions. The basic components of rules are therefore actions, and conditions, and their form, that bears a significant resemblance to the stimulus-reaction pairs of the behaviorist past, is as follows «*IF condition, THEN action.*» Condition-action rules (sometimes called «production rules» or «classifiers») underlie much work in AI, from the classic work of Newell and Simon (1972), to the large array of work in Expert System, and other, related approaches (e.g., Klahr, Langley, and Neeches, 1987; Holland, Holyoak, Nisbett and Thagard, 1986; Holland, 1986).

Those systems are based on a set of facts (both about the environment and their internal states). The rules (or some subset of them) are compared against the set of facts (or some subset of these). If the condition encoded in the rule is fulfilled, the action will be carried out by the system.

The activity of a production-based system may be decomposed in cycles. The first part of the cycle consists in identifying a set of rules that have their conditions satisfied. Next, a subset of these rules are selected, by some principle that varies from one system to the next. Finally, the selected rules are «fired», i.e. the

action part in them is effected. Some of the actions are overt, others consist in writing an internal message. The following cycle then begins, and takes into account changes in the facts at the system's disposal, both changes in the environment and internal changes brought about by the previous cycle.

Such rules system present two major advantages. First, the «language» of production rules is a universal programming language, that is, it is in principle possible to program with it any procedure that can be programmed at all (Laird, Newell and Rosenbloom, 1987). Second, it is «modular»: its elements are rules that can be identified, described, and their specific contribution to the overall efficacy of the system can be understood in relative isolation.

Expert systems

Probably the best-known rule-based systems are those called «Expert systems» or «knowledge-based systems» (David, Krivine, and Simmons, 1993). Expert systems are computerized system predicated on the notion that expertise consists largely in specialized knowledge, whereas the skill of combining existing knowledge with the specifics of the case at hand does not change much with increased expertise. «Knowledge is power,» at the slogan goes. It makes therefore sense to device so-called «inference engines», pure reasoning mechanisms. These are then given provided with the experts' knowledge, under the form of a set of rules, and specific information on which the expertise is to be brought to bear. There is therefore a separation of knowledge and treatment.

Knowledge is represented by production rules, as for example, in a medical expert system:

IF *(complaint=headache* AND *time-frame=acute* AND *intensity=serious)*
THEN ASSERT *(probability_bacterial_infection = .4; probability_stroke=.5)*

There exist a small set of control systems that may be used to combine such rules and the specific facts. The control systems are data-driven or hypothesis-driven in varying proportions. The system collects more information, integrates it into probabilistic arguments until an overall diagnosis or conclusion is reached. Expert systems have been built for a wide array of purposes, and specialized tools for constructing them rapidly are commercially available. Production systems, then, constitute a class of effective representational systems. While such rules can describe states of understanding, this by itself does not specify how a system might, on its own, develop a set of suitable rules to represent the structure of a knowledge domain. Can the process of constructing a model of expertise be supported by machine learning techniques?

The *Traveler* model (Leiser, 1987; Leiser and Zilbershatz, 1989) constitutes a simple example. This program is based on simple rules, telling it how proceed along its route. All rules are of the form: *IF (you are at location X)* AND *(your goal is location Y), THEN (perform action A)*. The system «travels» along routes, progressively discovering, in its own database, links between an increasing num-

ber of places. At first, the Traveler only links nearby routes. As its experience increases, there grows a complex, integrated system of main and subsidiary routes that carry the mark of the systems' specific developmental history. From a small set of local links, emerges progressively a large system of interrelated routes.

In a similar vein, models such as ACT (Anderson,1983,1986) and SOAR (Laird, Newell and Rosenbloom, 1987) are based on the refinement of production rules, and links between existing concepts. The method is homogeneous: concepts and production rules belong to different substantive domains are not formally defined by the system such. Yet, this homogeneous system may end up with several domains, in the sense that activation of individual concepts in one domain lead consistently to the activation of the other concepts in the same domain, if past experience was such that it linked certain concepts and production rules together in an associative web. Semantic fields are emergent with respect to this representation, as were main routes in the previous example. The construction and structuring of domains does not result from the application of some powerful generalization mechanism to a large data-base, but rather is the cumulative outcome of many specific past experiences.

Not all rules whose conditions are satisfied can actually perform the associated actions. Rules compete, and different systems designers have suggested a range of such «conflict resolution» principles. Existing rules can be modified in various ways to improve the efficiency of the system. Anderson and his collaborators (Lewis, 1987; Anderson, 1990; Klahr, Langley and Neeches, 1987) describe several mechanisms to effect this. Some of these really do not make any principle difference, since the outcome is merely to render knowledge more efficient, but without changing its nature or organization. Others change the rules themselves, and serve to make the system more in tune with its environment. Several improvement processes were described:

- *Compilation* takes several pieces of information, some of which are not rules at all, and combines them into a single, smooth, automatic rule. In particular, it includes *composition*, that combines several productions into one, and *proceduralization*, that collapses the actions of information retrieval from declarative memory and production matching, a feat into which we cannot go here.
- *Generalization*: this inductive mechanism takes several productions and creates a new one, trying to achieve a maximal common generalization of the initial rules. In so doing, constants are replaced by variables, while restrictive conditions that are not shared by the procedures are removed. The resulting production must then prove its worth in competition with others.
- *Discrimination*: this is the contrary of generalization: a successful general rule may have conditions added, on the basis of other rules that worked well: restrictive conditions existing in other productions are copied.

Genetic Algorithm

The improvement processes we just saw are all reasonable enough, but one would feel unhappy at the wise guiding hand that takes rules and delibera-

tely tries to improve them, rather like an engineer might try to combine two good design principles. The processes of discovering and improving rules have also been studied by an entirely different form of system, called the *Genetic Algorithm.*[1] In its best-know form, the Genetic Algorithm works on classifier systems, which are parallel, message-passing, rule-based systems wherein all rules have the same simple form. The condition part specifies what kind of messages satisfy the rule, and the action part specifies what message is to be sent when the rule is satisfied.

Classifier system consists of four basic parts: The input interface, that translates the current state of the environment into messages; the «classifier rules», the rules used by the system, define the actions done by the system in processing every kind of message; the message list, that contains all the current messages; and the output interface, that translates some messages into effector actions.

A classifier system basic cycle consists of the same steps outlined above for rule-based systems in general: add all messages from the interface to the message list, compare all messages on the list to all conditions of all rules, and record all satisfied conditions. For each of them, post the message specified by its action part; replace all messages on the list by the list of new messages; translate messages to requirements on the interface, producing the system's current output. In that sense, the classifier system is a parallel system, since many rules are applied simultaneously, before the next cycle.

The genetic algorithm (GA) works on such «populations» of rules. The algorithm is explicitly modeled upon Darwin's theory of evolution, and population genetics' analysis of the point of sexual reproduction. This parentage is very natural, since it is the same effect is being sought: the improved adaptation of small elements, without any central omniscient agency responsible for making intelligent progress. Neither Darwin species nor the classifier populations know which way it is best to evolve, and both try to improve the population's adaptive success in an environment. To recall, Darwin's theory is based on the notion that different individuals in a population will be individually more or less fit. By asexual reproduction, the fitter individuals will have a larger number of offspring, since they are by definition those who survive better than others. The next generation will therefore have a better proportion of fit individuals. Sexual reproduction goes one important step further: offspring recombine properties of parents. As a consequence, some of those recombinations, the luckier ones, may be better adapted than parents individually.

GA similarly works on populations. It selects good classifier rules as parents, forming offspring by recombining components from the parent classifiers. The offspring displaces weak classifiers and enter into competition, being activated and tested when their conditions are satisfied. More specifically, the GA operator selects good rules, then recombines them. It keeps therefore the same number of rules, but increases the proportion of those which have strong, «fit»

1. The «genetic algorithm» may also be used for entirely different functions, see below for an example.

building blocks. The cycle consists simply in selecting pairs of classifiers, with a probability proportional to the strength of the rules, then applying the cross-over operation, modeled on the chromosomal crossover: it exchange random continuous corresponding segments of the condition or of the message parts, and so creates a new pair of classifiers. This pair then replaces another pair of classifiers, selected for their low strength.

Credit Assignment

Of course, this scheme supposes that rules are attributed some measure of fitness, of «strength». Several schemes have been devised, based on various possible measures of past effectiveness. One of the better known is called the *Bucket brigade algorithm*. We will briefly discuss it here because it tackles an important difficulty of rule-based system, namely the problem of credit assignment. The problem is the following: rules should be somehow reinforced for being successful. But most rules do not have any overt action, and reinforcement is rare. The system's behavior is mostly stage setting, that makes later successes possible. The problem is especially difficult for parallel systems, where only some of the rules active at a given time may be instrumental in attaining success, and credit assignment should nevertheless reinforce, on balance, those rules that were effective.

Finally, the effectiveness of rules is not a property of the individual rules taken singly. While rules, as we described earlier, are modular in the sense that it is usually possible to describe the contribution of each rule to the system, it is nevertheless true that the effectiveness of a given rules depends wholly on the presence and activity of other rules. Reinforcement should therefore be dependent on the strength of other rules.

Young (1976), for instance, studied the acquisition of seriation procedures by children, and analyzed it as the construction of production systems to effect the task. He identified «seriation kits», sets of possible rules from which the production rules are to be selected. Effective procedures, for his task, must exhibit a good balance between evaluation, correction and selection. If one of the dimensions is weak, it can be compensated by the other two. If a proposed procedure is weak in all three dimensions, it just won't work. According to Young, children will generate just about any combination of rules, and some combination will work out right. Development is seen by him as locally meaningful: the coherence of the procedure as a whole does not require particular consideration. For him, production systems have no psychological existence as such. Procedures are seen as nothing more than a transient collaboration of individual rules. Actually, the subject does not only choose individual helpings (rules) to form a dinner out of a single menu, to use Young's metaphor. He also selects the menu itself. Children possess several menus, and the various seriation kits out of which procedures may be constructed correspond to basic strategies. It can be shown that the individual child knows several such basic strategies (Leiser and Gilliéron, 1990). Indeed, Young himself lead children to switch strategies in accor-

dance with the particular task used (successive presentation of the sticks by the Experimenter, screen, etc.). Since procedures are combined out of rules in a single set, the kits form consistent wholes, whose unity is psychologically real. I submit that the claimed independence of the rules is an illusion. Their evolutionary history was communal, and their success a joint result. The rules evolve as coherent systems of co-adapted elements. True, the child need not be aware of the overall coherence. He must merely replace a rule when the result is unsatisfactory. But the solidarity of the set of rules is nonetheless an evolutionary result, and Young's findings and analyzes may be seen as founding a taxonomy of viable *combinations* of rules.

It is these problems that the *Bucket Brigade Algorithm* (BBA) is designed to solve. To implement the algorithm, each classifier is assigned a quantity called its strength. The BBA adjusts the strength to reflect the classifier's overall contribution to the system effectiveness. The strength is then used as the basis of a competition. Each step, every classifier whose conditions are fulfilled makes a «bid» based on this strength. The higher bidders have their messages written on the message list for the next step.[2]

The height of the bid depends on the strength of the rule (that reflects past usefulness) and on other aspects that we will ignore here. Following a successful bid, i.e., if the rule is actually selected to have its message written on the message list, the strength of the rule is *diminished* by the size of its bid, and the bid itself is distributed among those rules that contributed in the previous step to this rule being selected. The operation of the BBA can be understood by an economic analogy: each rule is a kind of middleman in the economy. Each middleman deals only with its suppliers, the rules posting messages satisfying its conditions, and which its customers, the rules with conditions satisfied by the message sent by the middleman. Whenever a rule wins a bidding competition, it pays out part of its strength to its suppliers. If it did not with the competition, it pays nothing. As one of the winners of the competition, the rule becomes active, serving as supplier to its customers, and receiving payment from them in turn, if it is at all useful. The rule's strength measure its ability to turn, on average, a profit. If a rule receives more from its customers than it paid out, it has made a profit, and its strength is increased.

If the customers are on average profitable, i.e., effective, they will have high strength, and accordingly pay out to their supplier, who will become stronger. The profitability of a rule depends upon their being integrated into sequences leading to ultimately profitable consumers, that is, that eventually receive payment, reinforcement from the environment for success in it. In this way, the bucket brigade algorithm ensures that part of the ultimate payment, eventually, finds its way also to the early stage setting activity. This algorithm reinforces cooperation, hierarchical planning, and indeed, any form of organization susceptible to increase ultimate success. In addition to the GA described above, *mutations* may be used as well. This is important since the GA only combines existing seg-

2. Actually, the size of the bid only determines the *probability* that the message will be written, so that lower bidder also have the occasional chance to prove their value.

ments of existing rules, and this means that any radically different component cannot enter into the competition and prove its values. Mutation, i.e., the random exchange of a segment by another, comes to fill this need.

Overall, then, the regular functioning cycle of the classifier system, the bucket brigade algorithm that updates strengths advisedly, and the genetic algorithm that searches for better combinations creates all three types of rules modifications: generalization, specialization, and novelty.

Evaluating the Approach

There are serious doubts as to the true effectiveness of this approach. Significantly, these are coming even from that most pragmatic of Artificial Intelligence research domains, knowledge engineering (Schreiber, Wieling and Breuker, 1993), the branch of applied AI concerned with creating effective expert system. The basic approach to develop expert system has traditionally been via a process called «knowledge acquisition», in the course of which a –human– expert attempts to define the rules he or she uses in coming to a decision. These rules are painstakingly elicited by someone familiar with the programming language to be used, encoded, and these form the basis of the rules. Knowledge acquired from the expert was immediately implemented using a knowledge representation formalism. The underlying assumption was that frames or production rules represent knowledge identical to the cognitive foundation of human expertise.

Knowledge acquisition is no longer viewed as a process which directly transfers knowledge from a human to an implemented computer program but rather as a modeling process. The result of knowledge acquisition is no longer only a running program but a set of models. One of these models describes the task which should be solved by the knowledge-based system and the knowledge which is required to solve the task effectively and efficiently. Both are described in an implementation- independent manner. Both the human expert and the implemented systems are specific instantiation of this general model. Two requirements are now recognized by the knowledge acquisition experts. First, the separation of the symbol level and the knowledge level: At the knowledge level, the expertise is described in an implementation-independent manner. It is described in terms of goals, operations, and knowledge about the relationships of goals and operations. At the symbol level, a specific computational agent is implemented which carries out the problem-solving process by means of a computer program. A symbol level description corresponds to an implementation or design specification.

Second, different modeling primitives are required for epistemologically different types of knowledge: A model of expertise contains different types of knowledge. Most approaches distinguish between domain knowledge, inference knowledge, and task-specific control knowledge. A further type of knowledge concerns the use of domain knowledge by the inference and control knowledge. Therefore, a model of expertise must explicitly distinguish between different types of knowledge and several modeling primitives must be defined for every

type as each type includes different knowledge entities. It follows that it is a mistake to limit learning to rules only. Construction of knowledge must take place at several epistemologically and functionally different levels, if it is to be effective (Dietterich, 1986). A simple example of this approach is «Explanation-based learning» (Minton *et al.*, 1990). EBL works by explaining why a particular example is an instance of a concept. The explanations are then converted into operational recognition rules. Specifically, EBL acquires search control rules. These are domain-specific, based on successful problem solving decision, major failures and unforeseen goal interactions. There are other examples, but none that does justice to the rich variety of higher-order structures (see Leiser, 1995, for a review of these questions).

We studied elsewhere (Leiser and Gillièron, 1990) at some length the close relations there are between the knowledge level, the various strata alluded to in the previous paragraph, and the acquisition of cognitive and epistemological structure. This requires identifying the *loci* where knowledge is found or shaped, more often than not in implicit form, and where it can evolve progressively, even if the underlying representation takes the form of explicit symbolic rules. Putative cognitive structures have very different scopes, and the way to conceive of them, and hence to model them is correspondingly different. Let us arbitrarily separate them into small- and large scope structures.

For the smaller scope, knowledge is implicit in data and associated procedures, in the representational formats, and in the system's architecture. One important part of development could be the transfer of knowledge between these levels (Leiser, 1987; Moshman, 1990; Clark, 1992; Karmiloff Smith, 1993). For the larger scope, knowledge consists in the adequation and organization of categories of thought (Kant, Piaget), ontogenies (Keil 1979), inborn naive theories (Carey, 1975) and the ways and conditions for differentiating or extending existing ones.

In a case study of seriation procedures (Leiser and Gillièron, 1990) we showed that the loci of logic are multiple and heterogeneous. Logic is found in rules for symbolic transformations, in the overall coherence of the procedure, in the ability to assimilate a given problem to an appropriate approach, and in the accommodatory potential of the procedure. In more detail:

Symbolic representation: Some simple inferences are readily described in terms of aspects of mental manipulation of symbolic representations. Thus, elementary reasoning steps may be directly based on basic logical rules. This is the oldest approach to the simulation of reasoning, going back to the notion of uniform resolution algorithms. The elementary rules accounting for the transformational steps may also be more specific to the domain at hand: «General methods are weak methods».

Overall coherence: The overall coherence of a procedure embodies another aspect of operative knowledge. While it is true that elementary production rules, say, may have their own, modular function, they co-evolved jointly under adaptive pressure, and their adaptation to a given task is a joint property. Moreover, the entire adaptive package may be transferred to an isomorphic domain, showing that its coherence may properly be ascribed to the system.

This is very clear in Genetic Algorithms (Holland *et al.*, 1986; Koza, 1991; Belew *et al.*, 1991; Boers *et al.*, 1993), by far the most successful model for the unguided development of production systems. Reinforcement of successful rules is calculated to reinforce successful cooperation between existing rules, rather than merely successful individual ones (as in Anderson, 1986; Klahr, Langley and Neeches, 1987).

Assimilation: Assimilation is a basic form of understanding. The use of a given representational format, procedure or architecture betrays or manifest an implicit belief in its appropriateness. The contents of this belief can be made explicit, and this is routinely done in software engineering, whenever an orderly attempt is made to define generic problem-solving methods, with an eye on reusing them on a wider range of problems (David, Krivine and Simmons, 1993). An essential aspect of development may be the exploitation of regularities in the data encoded at the symbolic level, to identify existing patterns, or even to construct representational formats that will embody those regularities in their very architecture. Such a passage could involve the extraction of the regularities as the basis for building a new representational architecture, by some identifiable process of reflective abstraction or rely on a distributed representation, where the distinction between the representation of rules and that of exemplars becomes moot (Shultz, Schmidt, Buckingham, and Mareshal, 1993).

Potential for Accommodation: The last and most subtle of the *loci* is the accommodation potential of the assimilatory schema, representational format, or architecture. The use of any of these structures betrays an implicit belief in its appropriateness, hence a specification of the range of application of the structure. However, an assimilatory structure may evolve, under influence of the occasions on which it was used. There is no telling which way it will drift, since this will depend on the history of its encounters with related problems. But it will not drift without restraint, and its evolution must accord with more abstract structural constraints.

Summarizing, higher order entities are an important component of a well-balanced account of cognitive development, whether in the machine or in humans.The relations between regularity patterns in the structure of already encoded data, and the subsequent ease of encoding of additional data obeying the same structural relationships, form a conceptually identifiable *locus* of knowledge construction.

II Connectionist systems

In discussing rules-based system, we sketched how rules may evolve progressively, even without a wise guiding hand, merely from interacting with the environment, by virtue of a suitable system architecture. However, rules-based system start with a strong ontological *a priori*: the set of properties and relations used to define the conditions and actions themselves.

The second class of learning systems we will discuss are variously called Parallel Distributed Processing, Neural Networks, and Connectionist systems.

These are associative systems, and go even further in circumventing the use explicit theories, since they make do without rules altogether. Even such concepts as they use are progressively constructed, the cumulative outcome of many interactions with the environment. It is this property that makes them such exciting architecture for studying the emergence of adaptive cognitive systems.

The least sophisticated of AI algorithms simulates the function of a collection of biological neuron. First proposed decades ago, the outline of the basic model is as follows. Input and output cells are interlinked, either directly or via a series of intermediate, «hidden» nodes. The links can vary over time. To represent these variable connections in our computer model we assign a multiplicative weight to each input pathway. A high weight term denotes a favorable connection. An output signal will be produced as a function of the sum of inputs to the neuron. The mechanism that allows us the simulate learning in our computer model consists in making adjustments to the weights to reinforce correct decisions and discourage incorrect decisions.

Notation

In analyzing learning by computational means, it is customary to simplify the treatment, without loss of generality, by using a mathematical convention. I will present it here before embarking on the substantive presentation. All the information to be learned is represented by simple vectors, composed exclusively of 0 and 1. It is important to realize that nothing much is lost by doing so, at least in principle. As is of course well known, modern digital computers all use a binary language internally. That approach is used in surveys, or any database or statistical computer program. A given position in an array may represent the property: «is male», or «agrees to the statement expressed in line 44 in the questionnaire». Inasmuch as we are interested, not in the *qualia*, the properties themselves, but in learning, displaying, analyzing relations between such properties, nothing is lost by transformation. If more precision is required, several positions in the vector may be used. It is generally accepted that any information that can be represented at all can be so represented, to an arbitrary degree of precision using this approach. For instance, assuming we want to represent colors, using that binary language, we can represent 2 colors, using only one slot, or four colors, if we use two, and in general $2n$ colors if we devote n slots to represent the specific color.

Supervised learning

The variety of approaches in this domain is extremely large (Chauvin and Rumelhart, 1995) and it is impossible to even give a feeling for the range. The general approach is as follows. Certain nodes of the network are defined explicitly as input, others as output. These comunicate to the outside the system's response. That output is evaluated, and the result of that evaluation is fed back to the

system to improve its behavior, so that next time its chances of producing a behavior with better evaluation will be improved. The nature of the evaluation varies from one system to the other. It can be specific, and based on comparisons between the actual output and the desired output. The feedback system can be qualitative only, informing the system on whether there is improvement or deterioration of performance.

This approach has been shown to be remarkably effective on a variety of tasks. The simplest of the algorithm links directly the input and the output elements, and searches for the best set of weights to enable the system to associate a set of inputs with a set of outputs. Other algorithm include one, or several intermediate layers (the *Backprop Algorithm*), that enable the linkage to be more complex, and indeed to develop some level of representation to mediate between input and outputs. Error messages from the environment is propagated back orderly via several layers, and the information obtained is used throughout the system.

Yet additional algorithms can recruit new elements to the intermediate layer, in order to construct a better representation. This notion is used by several otherwise diverse approaches. One approach, for instance, is to introduce hidden units one by one; each unit comes to account for the main remaining source of variance. Only when there is no further progress is the next hidden unit introduced (the *Cascade Correlation* algorithm, Fahlman and Lebiere,1990; Fahlman, 1991; Hoefeld and Fahlman, 1992). Is this way, an optimal number of hidden units is used. The point is that if the network has too few units, it will not behave the way required. But if it has too many, its effectiveness will be spread pointlessly across too many units, and the system will be overfitted: generalization will not be easy. Shultz *et al.* (1994) applied successfully this approach to various learning task, for example that of weight balancing.

Unsupervised learning: Self-organizing systems

In this class of systems, there is no external evaluation of the output. Learning can nevertheless proceed, thanks to an organizing principle, of which we shall see two. The *Self-Organizing Maps*, developed by Kohonen (1995, for a recent overview), are based on the formation of regular representations under localization constraint. And the *Auto-Associators*, that structure themselves according to a principle of «energy minimization.» We shall consider these approaches in turn.

Self-Organizing Maps

We just described the goal of the self-organizing maps (SOM) as the formation of regular representations under localization constraint. The system is composed of an input layer and a treatment layer. By regularity, we mean that similar information, similar input patterns, should be treated in a similar way. As

usual with connectionist models, two phases are distinguished in the functioning of the system. A learning phase, adaptation of the weight system. And an exploitation phase, during which the system displays what is has learned. The goal then is that during the exploitation phase, following presentation of an input vector, the set of activated units in the treatment layer is small, and restricted to only some of the treatment units. Localization here is to be understood in the wide sense. In the narrow sense, every input would leads to a single activated neuron, and this would simply be a classification. The Kohonen net are localized in the wide sense: the localization is a the level of a group of neighbors; the activation pattern is that of a single node with maximum activation, and decreasing levels of activation in its the neighborhood.

The learning principle involve two components: *competition*: forcing several response nodes to compete, and giving the advantage to the best one; and *specialization*: a learning law that biases competition towards those nodes that won on earlier occasions. More specifically, Kohonen compared the weights of all output nodes and picked the set of nodes having weights that closely matched the magnitude of the input signal. These two principles together form a feedback loop, that eventually lead to localization in the wide sense just described.

The self-organizing network consists of a matrix of output nodes j all of which are connected to every input node i. The algorithm determines the «winning» node j* that most closely matches the expected output as determined by the set of input nodes i. Modifying the weights of j* and its neighbors will produce a set of desired outcomes. More explicitly, the steps are as follows:

1. Presentation of the input vector. $X=x_1 \ldots x_n$.

2. Competition between the treatment units. The one whose weight vector best matches the unit vector is selected. This involves two subsets: (A) *Compute distance*: compute distance d_j between input node i and each output node j. (B) *Identify best match*: Select minimum distance and denote output node j with minimum d_j to be j*. Neighboring nodes will also be identified, according to their proximity to j*.

3. *Learning*: update weights for node j* and its neighbors: $w_{ij} = w_{ij} + e(x_i - w_{ij})Y_j$. The e term is used to control the rate of weight adjustment, Y_j is the neighborhood distance to j*. Observe that two distinct distance functions are used: d_j, the distance between input vector and weight vector, and Y_j for specifying proximity of nodes in the treatment layer.

In the course of learning, stabilization of learning is achieved by modulating the neighborhood function Y (neighborhoods are large at first, then decrease) and decreasing e at the same time.

Input vectors are presented successively, and each presentation leaves a trace, a residue that reflects their structure. The net result of the activity is that regions come to be defined in the treatment layer, and their topological organization reflects the regularity patterns in the input vectors. The Self Organizing Map is a «nonlinear projection» of the probability density function of the high-dimensional input data onto the treatment space. A typical application such SOM is in the analysis of complex vectorial data. But such maps, that enable one to visualize metric ordering relations of input samples and to find clusters in the in-

put data, are not only useful when they are examined by say humans. This organization can then be exploited by other modules, to form the basis of effective behavior, or indeed, for extracting their structure. We describe elsewhere (Leiser, 1994) a way to extract the very structure of inclusion relation from discovery of the relations entertained by different categories. In this way, Piaget's pseudo-operational abstraction may be implemented, and the following seeming paradox may be resolved: how can a scheme structure incoming data at the same time it is itself in the process of formation. SOM and other unsupervised learning methods show that progressive structuring can take place without overall understanding, as the residue of numerous encounters with data. But later, the pattern of information implicit in the data can be extracted, and form the basis of a better representation (Leiser, 1987; Karmiloff Smith, 1992; 1993).

The Auto-Associator

Auto-associators are another species of unsupervised learning. We saw earlier that in supervised learning, the algorithm links the input and the output elements, and searches for the best set of weights to enable the system to associate a set of inputs with a set of outputs, either directly or several intermediate layers. An interesting variant is when the input and output spaces are the same, and the output is looped back to the input. Under suitable conditions, the system will reverberated for a while, and eventually settle down in one «equilibrated» state. Intuitively, we may think of a surface, with certain dips –these are the attractor states, which may be numerous or not, close or far from one another. Deposit the system in the vicinity of one of them, with some friction, and it will slowly inch its way to one of the dips.

The goal of the learning mechanism is to transform certain states, defined on the basis of external reality, into such attractor states. Under certain conditions it is possible to define an «energy» function, and let the system converge to the closest *local* minimum of the function. By presenting many members of the same category, it can then be arranged that a state that correspond to the prototypical member of the category becomes an attractor. Present any member of the category to the system, and it will eventually reach the state corresponding to the prototypical member of the closest category, and this will be the case even if the prototype itself was never presented. Summarizing, there are two phases. The first involves «molding» the landscape, so that goals we want to achieve, e.g., existing categories with their labels, correspond to local minima. The second phase exploits the first one, for classification for instance: one presents a pattern, and watch it settle to the closest goal.

Another possibility is to search for the *absolute* minimum. This requires getting out of local minima. A method called «simulated annealing» controls the probabilities of going from one step to another, in the course of search: at first, local minima are no barrier to transition, and later they become so. This approach increases the chance of ending in the global minima of the function. An application of this mode of functioning is to use the network as a *pattern*

completion device: present part of the pattern, and it will settle into the complete one, making by the same token available all the information that correspond to it.

Epistemology and Architecture

In all those examples of connectionist learning, knowledge states are patterns of activation, rather than sets of propositions. Long-term knowledge is distributed amongst the weights connecting a large numbers of nodes. Retrieval is the reinstatement of a prior pattern of activation. Memory traces are simulated by changes in the weights connecting the nodes.

Can one separate a distinct knowledge level in the case of a neural network that performs a task successfully? In some cases, this is fairly straightforward. The *Forward Modeling* model of Jordan and Rumelhart (1992), for instance, is based on the construction of an internal model of the environment, that is used to devise the mapping between intentions and actions. But in other cases, things are far less clear: the overt organization of a neural network may not betray at all its functional organization. For example, the behavior of the dual-route model of Coltheart was recently reproduced by a system that has only a single route of processing. Coltheart (1994) argues that this exemplary conflict between his own psychological research and successes in modeling the same phenomenon by means of neural networks, is to be resolved by contrasting network architecture and functional architecture, and not letting the former distract one from studying the latter : «The network architecture is determined by the modelers, and is obvious. The functional architecture (...) can be very difficult to discover in realistically large networks. What are cognitive psychologists interested in? Well, obviously, functional architecture (p.21)».

Others view the two architectures as complementary (Bates and Elman, 1992; Plunkett and Sinha, 1992; Clark, 1992). To ground the attribution of a structure to a subject, one must show how development necessarily causes the structure to emerge, and conversely, why the structure is an attractor for the developmental process, as expressed in Piaget's dictum *Pas de structure sans genèse, pas de genèse sans structure* (no structure without development, no development without structure). A complete explanation would therefore involve all three terms : the emergent structure, the architecture of the learning mechanism, and a chronicle of the development. The structure sets the goal, with reference to the system's internal or external adaptation. The architecture accounts for how the system comes to construct the structure, and thereby explains the chronicle. A very similar view was put forward by Seidenberg and McClelland (1992; see also Plaut, McClelland, Seidenberg and Patterson, 1994) who describe their approach as follows : they postulate a specific way of representing the stimulus world, a learning rule, and an architecture, then observe that under those conditions, a certain type of knowledge representation *necessarily* results. It is not built in, but arises from interactions among independent factors and this, to the extent it succeeds, provides the explanation for

development. The study of emergent developmental properties and of the course of this emergence precedes the modeling work, whose task is to show why certain structures are acquired.

On Hybrid Systems

The models we have described are general principles, and in their implementations several principles may be combined. There are by now very many algorithms, whose properties are being studied analytically and by computer simulation. Inevitably, the question arises of how to combine the advantages of two or more mechanisms. Schyns (1991) advocates *modularity*, that is, the assignment of the primitives of a functional architecture onto different structural modules. The modules may be assembled hierarchically so that, as in his system, unsupervised learning achieves most of the preprocessing of the input patterns, whereas supervised learning then takes the output of that preprocessing to fulfill other tasks. Specifically, his system uses a SOM model to categorize the input, and a supervised auto-associator to learn labels for those categories. It can then be shown that the existence of a naming module, suitably connected, facilitates in various ways the establishment of an effective classification of the input. This analysis had been previously proposed for children learning of language: their conceptual repertoire must be advanced enough before they can learn language, but language in turn affects the organization of the conceptual categories.

Extensive work is currently taking place, trying to link the product of the massive, parallel based information processing of connectionist systems, with the more traditional and well-grounded symbolic based systems. We choose to present here a recent way of linking the two called *Evolutionary programming* (Kinnear, 1994, Koza, 1994). It is not widely known, nor is it likely to be used by individuals, but it shows in striking manner the room there remains for novel ideas in this domain.

Many seemingly different problems in artificial intelligence, symbolic processing, and machine learning can be viewed as requiring discovery of a computer program that produces some desired output for particular inputs. When viewed in this way, the process of solving these problems becomes equivalent to searching a space of possible computer programs for a most fit individual computer program. «Genetic programming» provides a way to search for this most fit individual computer program. In this paradigm, *populations* of computer programs are genetically bred using the Darwinian principle of survival of the fittest and using a genetic crossover (recombination) operator appropriate for genetically mating computer programs.

Examples come from the areas of planning, sequence induction, empirical discovery, solving equations, concept formation; automatic programming; pattern recognition. Since the Genetic Algorithm, with the cross-over function, combines and identifies useful components that form part of the evolved programs, it can be said to extract useful subroutines, i.e., functional modules that

are relatively independent. This has great importance for problem-solving. It is often argued that the process of solving complex problems can be automated by first decomposing the problem into subproblems, then solving the presumably simpler subproblems, and then assembling the solutions to the subproblems into an overall solution to the original problem. The overall effort required to solve a problem can potentially be reduced to the extent that the decomposition process uncovers subproblems that are easier to solve and to the extent that regularities in the problem environment permit multiple use of the solutions to the subproblems. Conventional techniques of machine learning and artificial intelligence provide no effective means for automatically executing this three-step problem-solving process on a computer.

This process is automatically implemented by means of the technique of automatically defined functions in the context of genetic programming. Automatically defined functions enable genetic programming to define useful and reusable subroutines dynamically during a run. This new technique was applied to solving, or approximately solving, example problems from the fields of Boolean function learning, symbolic regression, control, pattern recognition, robotics, classification, and molecular biology. In each case, the problem is automatically decomposed into subproblems; the subproblems are automatically solved; and the solutions to the subproblems are automatically assembled into a solution to the original problem. Leverage accrues because genetic programming with automatically defined functions repeatedly uses the solutions to the subproblems in the assembly of the solution to the overall problem. Moreover, genetic programming with automatically defined functions produces solutions that are simpler and smaller than the solutions obtained without automatically defined functions.

Final Comments

In this survey, I strove to suggest the nature of the impressive work taking place in Artificial Intelligence, especially those involving ideas that are less well-known to psychologists. Starting from the simple conceptual description of the lower and higher stages of development, we showed those stages may be described as collection of rules, each a modular entity that may be independently acquired or modified. We then sketched some mechanisms that allow this progress to take place on its own. First, we indicated how this occurs with the guiding hand of a central mechanism responsible for making appropriate generalizations, compilations and so on. We then saw some mechanism that are unguided, and nevertheless show promise of achieving the same goal.

The other class of learning systems we discuss are even more autonomous, in that the conceptual building blocks themselves are constructed by the system, the cumulative outcome of many interactions with the environment. Here too, we saw two types of systems. In our exposition, we made use of the classic distinction between supervised and non-supervised learning. One would

be tempted say that the former are non-constructive, and the latter are. But this is really misleading: all cases of learning are adaptation under constraints, the real difference being that constraints may be more or less specific. The proper distinction relates therefore to the nature of the constraints. These may be case-based, i.e., concern specific cases, or more global: the network architecture, learning function, activation functions, etc. The supervised systems, while indeed constructing an internal representation on their own, are guided by a tutor that indicates to them, after each trial, how successful their behavior is. The difference between actual and required behavior serves to guide the system to the construction of a good representation. Finally, the unsupervised systems construct a representation under general architectural constraints, but without being guided by specific, case by case information.

The automatic creation of complex cognitive structure, by the interplay of diverse architectures and substantive domains, promises to give substance to Piaget's description of equilibration (Piaget, 1975, pp. 180-181). Piaget's account was explicitly at the level of his «functional invariants» as opposed to that of specific mechanism. The distinction is indeed valid, and the possibility that a variety of mechanisms may fulfill the same function is well know. But as long as *no* mechanism is forthcoming, the account lacks a key component. Today, do we have some understanding of what emerging, incomplete knowledge might look like, and we are in a much better posture than in the past to discuss mechanisms of emergence. At the same time, the proposed systems are yet to demonstrate that they are capable of generating the range of psychological structures that are found to emerge. This will be especially difficult to the extent that the grain of the elements relied on by the mechanisms is small, the scope of what those systems are trying to learn is large, and there are fewer inbuilt assumptions about the domain to be acquired.

Perhaps the aspect in which the least work has been done to date in AI, whether in the symbolic or subsymbolic approach, concerns the interaction between knowledge domains. Domains do not evolve in isolation. Learning a $n+1^{th}$ domain is different from learning the first n. And knowing several domains has its impact on the individual domain. A universal novice is a novice in a way a domain-specific novice is not, for learning previous domains may have had its effect.

Known domains can serve as the source of analogies, as templates in the construction of the new domain, once both are sufficiently structured to detect the relevance of an analogy (Goswami, 1991; Holyoak, Novick and Meltz, 1994; Gentner and Toupin, 1986; Gentner, Ratterman, and Forbus,1993; Brown, 1989). A domain or scheme can be more or less profoundly affected by a successful analogy that it supported. There is a continuum of influence, ranging from the transient and superficial to the long-lasting and profound. *Transfer* is the weakest link type. The original scheme remains as it was, and the cognitive structure, the scheme, is not lastingly modified by the encounter. There are but very few ideas about mechanisms that might effect this, but there are some beginnings, even when the structure is as diffuse as in a back-propagation network (e.g., Pratt, 1993). «*Schema abstraction*» is different. In principle, transfer is

only within a zone of mental elasticity, the assimilatory region of the scheme. That zone has fuzzy boundaries: increased effort manages to effect the transfer to more distant items. The implicit definition of what the scheme does evolves as that zone extends. The original paradigmatic case need not remain central or retain a special status, as it becomes subsumed under the more abstract scheme that evolves. When this process has proceeded far enough, one talks more naturally of abstraction. The ultimate end-result is when structurally equivalent problems are solved with equal ease.

Another possibility concerns *coordination of domains*, where the two domains evolved and became structured independently The same empirical situation may give rise to contradictions, to incoherence, if assimilated to different domains simultaneously. Conflict resolutions go from *demarcation* to complete *merging* with attendant conceptual change. The former leaves the two domains as they were, merely adding the ability to discriminate when either of them is appropriate. The latter may involve a complete reworking of the two domains involved. Piaget (1975) viewed the need to coordinate one's mental structures as one of the two great sources of development, the other being the need to adapt to the external environment. While nothing precludes in principle the development of AI techniques to perform such coordination, only the very first steps have been taken (e.g. Holyoak and Thagard, 1989, Sloman, 1993).

This survey has not examined the philosophical arguments for or against the *posssibility* of realizing a constructionist programme within one of the varieties of AI. Such arguments are best handled by professional philosophers. Instead, it presented some of the current accomplishment of the field, and an inkling of what may reasonably be expected in the future. Perhaps this approach is just as well. Some years ago, Minsky and Papert (1967) published seemingly definitive mathematical arguments, that were taken to demonstrate the necessary barrenness of the connectionist approach. Researchers duly eschewed that approach during the following decades, until, some twenty years later, an apparently minor modification of the connectionism mechanism enabled the approach to sidestep those discouraging conclusions. Without denying the usefulness of conceptual clarification philosophical analysis brings to a problem, I suspect a similar fate will befall the *a priori* arguments over how far AI is compatible with constructivism.

The general orientation towards constructivism has changed over the last few years, both in the symbolic and the sub-symbolic approach. The over-simple equation of cognitive structure and data structure (Leiser, 1987) has given way to a realization that there are layers of cognitive stucture that, while depending on data structures, cannot be represented explicitly while evolving. This assessment is of course devoid of theoretical significance –it could just as well be a mistaken conclusion– but it does reflect the cumulative experience of earnest practitioners engaged in modeling knowledge acquisition. To be sure, this awareness is only slowly being translated into effective programs, as is only natural: the enterprise is immensely difficult. But the AI field demonstrated impressive vitality and creativity, and the question is at last able to make fruitful contact with a constructivist position.

REFERENCES

Anderson, J. R. (1983). *The Architecture of Cognition*. New York: Freeman.

Anderson, J.R. (1986). Knowledge compilation: the general learning mechanism. In R.S Michalski., J.G. Carbonnell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* vol. 2. Palo Alto: Tioga Press.

Anderson, J.R. (1990). A theory of the origins of human knowledge In J.G. Carbonell (Ed.), *Machine Learning: Paradigms and methods*. Cambridge, MA: MIT Press

Anzai, Y. & Simon, H.A. (1979). The theory of learning by doing. *Psychological Review, 86*, 124-140.

Bates, E. A. & Elman, J. L. (1992). Connectionism and the study of change. In M.H. Johnson (Ed.), *Brain Development and Cognition: A reader*. Oxford: Blackwell.

Belew, R.K., McInerney & Schraudolph, N.N. (1991). Evolving Networks: Using the Genetic Algorithm with Connectionist Learning. In C.G. Langton, C. Taylor, J.D. Farmer and S. Rasmussen (Ed.), *Artificial Life II*, vol. X. New York: Addison-Wesley.

Boers, E.J.W., Kuiper, H. Happel, B.L.M. & Sprinkjuizen-Kuyper, I.G. (1993). *Designing modular artificial neural networks, 93-24*. Department of Computer Science Leiden University.

Brown, A.L. (1989). Analogical learning and transfer: What develops? In S. Vosniadou and A. Ortony (1989), *Similarity and analogical reasoning* (369-412). Cambridge, MA: Cambridge University Press.

Carbonell, J.G. (Ed.) (1990). *Machine Learning: Paradigms and methods*. Cambridge, MA: MIT Press.

Carey, S. (1985). *Conceptual change in childhood*. Cambridge, MA: MIT Press.

Chauvin, Y. and Rumelhart, D. (Eds.) (1995). *Back Propagation: Theory, architectures and applications*. Hillsdale, NJ: Erlbaum.

Clark, A. (1990). Connectionism, competence and explanation. *British Journal for the Philosophy of Science, 41*, 195-22.

Coltheart, M. (1994). Connectionist modelling and cognitive psychology. In J. Wiles, C. Latimer, and C. Stevens (Ed.), *Proceedings of Conference on Connectionist modelling and psychology*. Brisbane: Department of Psychology, University of Queensland.

David, J.M., Krivine, J.-P. and Simmons, R. (1993) (Eds.). *Second Generation Expert Systems*. Berlin: Springer.

Dietterich, T. G. (1986). Learning at the knowledge level. *Machine Learning, 1* (3), 287-316.

Dresher, G. L. (1991). *Made-up minds: a constructivist approach to Artificial Intelligence*. Cambridge, Mass: The MIT Press.

Fahlman, S.E. (1991) The Recurrent Cascade-Correlation Architecture. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky (Eds.), *Advances in Neural Information Processing Systems 3* (190-196). Los Altos CA: Morgan Kaufmann Publishers.

Fahlman, S.E. & Lebiere, Ch. (1990). «The Cascade-Correlation Learning Architecture». In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2* (524-532). Los Altos CA: Morgan Kaufmann Publishers.

Gentner, D., Ratterman, M.J. &. Forbus, K.D. (1993). The roles of similarity in transfer: separating retrievability from inferential soundness. *Cognitive Psychology, 12*, 306-355.

Goswami, U. (1991). Analogical reasoning: what develops? A review of research and theory. *Child Development, 62*, 1-22.

Hoehfeld, M. and Fahlman, S.E. (1992). Learning with Limited Numerical Precision Using the Cascade-Correlation Learning Algorithm, *IEEE Transactions on Neural Networks*, Vol. 3, 602-611.

Holland J.H. (1986). Escaping Brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence* Approach vol. II. Los Altos CA: Morgan Kaufmann.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction - Processes of Inference, Learning, and Discovery*. Cambridge, MA: MIT Press.

Holyoak, K.J. & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13*, 295-355.

Holyoak, K.J. Novick, L.R. & Meltz, E. (1994). Component processes in analogical transfer: mapping, pattern completion and adaptation. In K.J. Holyoak and J.A. Barnden (Ed.), *Advances in connectionist and neural computation theory*. Vol. 2: *Analogical connections* (113-180). Norwood, NJ: Ablex.

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: supervised learning with a distal teacher. *Cognitive Science, 16*, 307-354.

Karmiloff Smith, A. (1992). Nature, Nurture and pdp: Preposterous Developmental Postulates? *Connection Science, 4* (3), 253-269.

Karmiloff Smith, A. (1993). *Beyond modularity: a developmental perspective on cognitive science*. Cambridge, MA: MIT Press.

Keil, F.C. (1979). *Semantic and conceptual development: An ontological perspective*. Cambridge, MA: Cambridge University Press.

Kinnear, K. E., Jr. (1994) (Ed.). *Advances in Genetic Programming*. Cambridge, MA: MIT Press.

Klahr, D. Langley, P. & Neeches, R. (Eds.) (1987). *Production system models of learning and development*. Cambridge, MA: MIT Press.

Kohonen, T. (1995). Self-*Organizing Maps Springer Series in Information Sciences*, Vol. 30, Springer.

Koza, J.R. (1991). *The genetic programming paradigm: genetically breeding populations of computer programs to solve problems*. Computer Science Department, Stanford University.

Koza, J.R. (1994). *Genetic Programming II*. Cambridge, MA: MIT Press.

Laird, J. E, Newell, A.& Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. *Artificial Intelligence, 33*, 1-64.

Leiser, D. (1987). The changing relations of representation and cognitive structure during the development of a cognitive map. *New Ideas in Psychology, 5*, 95-11.

Leiser D. (1990). Evolution, development and learning in cognitive science. *Behavioral and Brain Sciences, 13* (1) 80-81.

Leiser, D. (1994). *Acquisition de la relation d'inclusion par PDP*. Unpublished manuscript. Université Pierre et Marie Curie, Paris.

Leiser, D. (1995). Computational emergence and developmental emergence: a sobering survey. *Proceedings of the ECCS'95 European Conference on Cognitive Science*, Saint Malo, France, April 1995

Leiser, D. & Zilbershatz, A. (1989). The Traveler - a computational model of spatial network learning. *Environment and Behavior, 21* (4), 435-463.

Leiser, D., & Gillièron, G. (1990). *Cognitive science and genetic epistemology - a case study of understanding*. New York: Plenum.

Lewis, C. (1987). Composition of productions. In D. Klahr, P. Langley, and R. Neeches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.

Minsky, M. & Papert, S. (1967). *Perceptrons*. Cambridge, MA: MIT Press.

Minton, S., Carbonell, J.G., Knoblock, C.A., Kuokka, R., Etzioni, O. & Gil, Y. (1990). Explanation-based learning: a problem solving perspective. In J.G. Carbonell (Ed.), *Machine Learning: Paradigms and methods*. Cambridge, MA: MIT Press.

Moshman, D. (1990). The development of metalogical understanding. In W.F. Overton (Ed.), *Reasoning, necessity and logic: developmental perspectives*. Hillsdale, NJ: Erlbaum.

Newell, A. & Simon, H. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.

Piaget, J. (1975). *L'Équilibration des structures cognitives: Problème central du développement*. Études d'Epistémologie Génétique, vol 33. Paris: Presses Universitaires de France.

Plaut, D. C., McClelland, J. L., Seidenberg, M. S.,& Patterson, K. E. (1994). *Understanding Normal and Impaired Word Reading: Computational Principles in Quasi-Regular Domains* (Technical Report No. PDP.CNS.94.5). Carnegie Mellon University.

Plunkett, K., & Sinha, C. (1992). Connectionism and developmental theory. *British Journal of Developmental Psychology, 10*, 209-254.

Pratt, L.Y. (1993). *Transfering previously learned back-propagation neural networks to new learning tasks*. Unpublished Doctoral Dissertation, Rutgers - The State University of New York.

Schreiber, G., Wielinga B. & Breuker, J. (1993) (Eds.). *KADS. A Principled Approach to Knowledge-Based System Development, Knowledge-Based Systems*, vol 11. London: Academic Press.

Schyns, P. G. (1991). A modular neural network model of concept acquisition. *Cognitive Science, 15*, 461-508.

Seidenberg, M. S. & McClelland, J. L. (1992). *Connectionist Models of Explanatory Theories in Cognition* (No. PDP.CNS.92.4). Department of Psychology, Carnegie Mellon University.

Shultz, T. R., Schmidt, W. C., Buckingham, D. & Mareshal, D. (1994). Modeling cognitive development with a generative connectionist algorithm. In T. Simon and G. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling*. Hillsdale, NJ: Erlbaum.

Sloman, S.A. (1993). Feature-based induction. *Cognitive Psychology, 25* (2), 231-280.

Young, R. M. (1976). *Children's seriation behavior - an Artificial Intelligence analysis of a Piagetian task*. Basel: Birkhäuser.