A Constructivist Model of Robot Perception and Performance

Joseph A. Lewis and George F. Luger
jalewis@cs.unm.edu, luger@cs.unm.edu
Department of Computer Science; FEC 323 University of New Mexico
Albuquerque, NM 87131

Abstract

We present a new architecture for robot control. It is rooted in notions from Brooks' subsumption architecture that have been extended to include an internal representation which matures as it experiences the world. The architecture is based on the Copycat program of Mitchell and Hofstadter, an early model of fluid representation whose details we discuss. We show how our architecture develops a representation of its environment through a continuing interaction with it. The architecture is founded on a dynamical systems interpretation of representation and demonstrates the importance of the use of "embodiment", reflecting a constructivist epistemology, in the control system of a robot designed to explore and utilize its environment in the context of problem solving.

Introduction

We present an architecture for controlling a robot based on the constructivist insight that representation occurs as a product of the active interpretation of perception-based experience. This architecture supports the control program for a robot whose task is to move about, explore, and map its world in the process of problem solving. The robot generates a representation of its environment by converting sequences of sensory data into perceived "objects". Our approach allows the robot to behave more robustly than does the use of the more traditional "preinterpreted" (McGonigle 1998) representations of its world. In this paper, we first describe the details of the model and then show its capacity to interactively construct a representation of surfaces and gaps (discovering the "objects") in its environment. We include in our presentation some of the philosophical and epistemological issues supporting our approach.

Our work builds on research from several disciplines. These include: behavior-based robotics (Brooks and Stein 1994), the "dynamical nature" of representation and intelligence (Steels 1995, 1996), and the philosophical insights of Maturana and Varela (1980) and Clark (1997), who emphasize the organization of living systems and their "coupling" with their environments. Further support for our approach comes from Holland's (1986, 1998) ideas on emergence in the context of classifier systems, and work on "fluid representations" in software architectures, for example Copycat, proposed by Mitchell and Hofstadter (Mitchell 1993).

Traditional cognitive science and artificial intelligence have focused on building the (supposedly static) structures involved in representational processes. The peculiar fluid quality of actual structures that support complex problem solving in changing environments has resisted elucidation. More recently a shift of focus, generated in part from the study of complex adaptive systems, has driven research to attempt to characterize the dynamical processes underlying these representational structures. Architectures whose representations are implicit in behavior, supported by dynamical constraints and triggers from the environment, have begun to validate the constructivist claim that "refinement of an interpretive framework is usually driven by the tension between the pattern of interpretation and the demands of successful interaction." (Luger 1994). These models also provide suitable tests for the assertion that representations only have meaning in the context of embedding experiences.

Our control architecture implicitly defines intelligence with the four characteristics of evolving complex adaptive systems proposed by Steels (1996). The first of these criteria is *self-maintenance* (we prefer the term *autopoeisis* from Maturana and Varela (1980) who also describe a "mutual maintenance" relationship among system components). The remaining criteria for describing intelligence are *adaptivity*, *information preservation*, and, in response to the demands of a complex environment, a *spontaneous increase in complexity*.

We also follow Steels (1996) suggestions that there are two ways that intelligent systems can achieve these four criteria. The first is through the use of a general purpose dynamical architecture. The second is through the capture of the emergent properties of interactive behavior, enabling the formation of concepts about and representations of the environment. We feel that the emergence of structures evolved through "coupling with" an environment is a defining feature of intelligence, and call this the *behaviorally coupled representation*. Furthermore, this "embodiment" is so critical to the study of intelligence that at least at the present state of our understanding, building and testing robots is an insightful necessity.

A New Architecture for Robot Control

Most early approaches to robotics subscribe to an implicit *sense-model-plan-act* framework (Brooks 1991b). In the 1980s, concern arose about the performance and complexity entailed by this framework when applied to adaptive autonomous agents functioning in actual environments.

Lewis, J.A. and Luger, G.F. "A Constructivist Model of Robot Perception and Performance". In *Proceedings of the Conference of the Cognitive Science Society*, Hillsdale NJ: Lawrence Earlbaum (2000).

This concern motivated a shift in thinking about the design of robotic systems as well as conjectures about the organization and use of intelligence itself.

The subsumption architecture (Brooks 1991a) marked the beginning of *behavior-based robotics*. Behavior-based robotics emphasizes the integration of semi-independent layers that produce behaviors directly from input rather than each contributing to a stage of the sense-model-plan-act framework. The focus is on interaction with the environment as a trigger for behavior rather than use of explicit representation. The ability to react to dynamic features of an unpredictable environment and to generate robust behavior despite sensor uncertainty is a signature of this behavior-based approach. Testing physically constructed robots interacting with complex worlds bears much weight in this new paradigm of robotics research. The behavior-based approach is a useful framework for organizing our understanding of intelligence (Brooks 1991b).

Brooks was right to criticize AI for the use of representational schemes with fixed and predetermined interpretations. As a result of moving away from the use of explicit representations, however, too little emphasis has been placed on the "appropriate" role of representation in intelligent problem solving. We want to pair Brooks' insights with a flexible representation that evolves with its interactions within an environment. A new dynamical model of representation, focusing on the role of emergent structure in behaviorally coupled systems, will accompany our new framework for robotics. McGonigle, referring to the polarity between representational stances, claims "we have the concept of a co-evolving agent and environment leading to a mutual specification..." (McGonigle 1998). To explore this new notion of representation, we must develop models that are both dynamical and embodied. Then we must seek mechanisms in those models for the emergence of structures coupled through system behavior to the environment.

Maturana identifies a hallmark of living systems which he calls *structural coupling* (Maturana and Varela 1980). Structural coupling means that the environment triggers changes in the internal structures of a system; but the nature of those changes is dictated by the dynamics of the system rather than being specified by the environment. An "embodied" model is one which participates in the dynamics of its world and which undergoes changes in its internal processes triggered by events in the environment. Representation for a robot control system can be achieved by providing a sufficiently rich dynamical system inside the robot to enable structural coupling to take place between the robot control architecture and the environment.

In spite of admonitions against representation, the use of partial world models may actually increase the ability of dynamical systems to meet the real-time demands of their environments. Clark discusses this in connection with Kawato's work on proprioception (Clark 1997). Partial models devoted to the improvement of specific behavior are called *niche models* (Clark 1997). Representations can be partial because they derive their meaning from the context of interactions within an environment.

"Fluid Representation" and Copycat

Copycat (Mitchell 1993) is one of the first computer programs to attempt to capture the dynamical processes from which symbolic or representation based behavior can emerge. Copycat solves analogy problems such as, if "abc" becomes "abd" what does "ijk" become? Such seemingly simple analogies involve evolving, context-dependent processes of integration and differentiation that are at the core of intelligent problem solving.

In addition to its novel mechanisms for parallelism and flexible adaptation, one of Copycat's most important components is the *slipnet*. The slipnet is a semantic network organized with spreading activation and multiple kinds of links among its nodes, some of which can change in length. The processes which evolve representational structure impact the topology of the slipnet, making the program's own behavior part of the adaptive control. For example, if several interacting processes have successfully built structures about *opposite* relationships among the input, the node for *opposite* in the slipnet becomes more active. Furthermore, *opposite* links become shorter and more likely to be traversed, and further processes to explore *opposite* are generated. Figure 1 shows the lengths of links between two nodes, *successor* and *predecessor*, as 85. This value shrinks as the label node for those links, namely *opposite*, gets an increase in activation (shown inside the ovals), making substitutions of one for the other more likely. In addition to spreading activation, this is the method by which slipnet evolves its meanings in response to events in its environment.

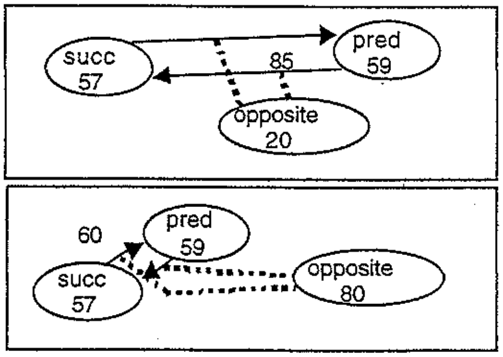

Figure 1: The Evoloving Slipnet

Interacting with the slipnet in Copycat are the *coderack* and the *workspace*. The interactions of these three components of Copycat are mediated by the system's *temperature*, which measures the cohesion of the workspace structure. The workspace is a global arena for creating structures that the other components of the system can inspect. In this sense it is much like a Blackboard (Luger and Stubblefield 1998) or the message area in Holland's (1986) classifier system. Copycat's coderack is a priority biased probabilistic queue containing codelets. Codelets are small pieces of executable code designed to interact with the objects in the workspace and to attempt to further some small part of the evolving solution, or even more simply, to explore different facets of the problem space. The codelets are very much like the individual classifiers in Holland's (1986) original system.

Copycat is a unique hybrid between serial and parallel execution, between goal-driven and data-driven search, and in particular between the symbolic and connectionist paradigms. The Copycat architecture models the fluid representation of concepts and their adaptive application to the active construction of features from perceived data.

One limitation of the Copycat program is that it has only one point of interaction with its environment (the initial exposure to the letter-string analogy problem). There are no means for continuing interaction with the external environment, only an ongoing maturation of the internal structures of the program guided by its own context-sensitive semantic network.

A second limitation of Copycat is the program's restricted domain. Even more importantly, the domain structure in Copycat, which facilitates exploration of fluid concepts in high-level perceptual processes, also restricts the interpretations available to the program of its developing representation. For example, the relationships possible between structures in Copycat, like *predecessor*, *successor*, and *opposite*, are derived from abstract ordering relationships in the alphabet. We have extended the program to include richer semantic relationships whose application can continue to evolve throughout the program's interaction with its environment. Related issues, for example, the ability to interactively discern new rules and interpretations from observed behavior, are addressed in the Metacat project (Marshall 1999). By using the ideas from Copycat and Metacat in our own embodied world of the robot, we have begun to address these limitations.

The Madcat Architecture

The Madcat project explores how an architecture similar to Copycat can be used to detect abstract features of sensory data obtained from an ongoing dialog with the environment. With its three mutually self-maintaining components, the slipnet, workspace and the coderack, the Copycat architecture is an autopoeitic system and a starting point for a general model of embodied intelligence. Copycat exhibits the characteristics of an evolving complex adaptive system relying on a subsymbolic dynamical system whose structural coupling supports its representation of a domain. In Madcat the emergence of representational structures is coupled to the environment through system behavior.

The Madcat project extends the Copycat architecture to the control system for a robot, producing a control architecture capable of ongoing interaction with a dynamic environment. The Madcat robot is a Nomad Super Scout II capable of translational and rotational motion with 6 bump sensors, 16 sonar sensors, and a color vision camera. This collaboration between Copycat and the Nomad robot produced the project name *Madcat*. The ultimate goal of our research is to construct a robot architecture that, from its emergent exploratory behavior, can build a flexible representation of its environment that improves its real-time performance.

In our research we look for behaviors that can be made more effective by niche models (Clark 1997). We build the individual components of the architecture and their rules to interact with data from the sensors and relationships among that data. The resulting emergent structures are correlated with the events in the environment, such as the passing of a corner. The internal "representations" of these events interact with the control system to produce behavior that is based on that "representation".

For example, we overcome certain sensor limitations in the robot using this emergent representation scheme. The maturation of the representation through interaction with the environment is what makes this feasible for a robot whose motion creates constant change in its sensory data. This evolving representation in the behavior-based framework is an important feature of this model.

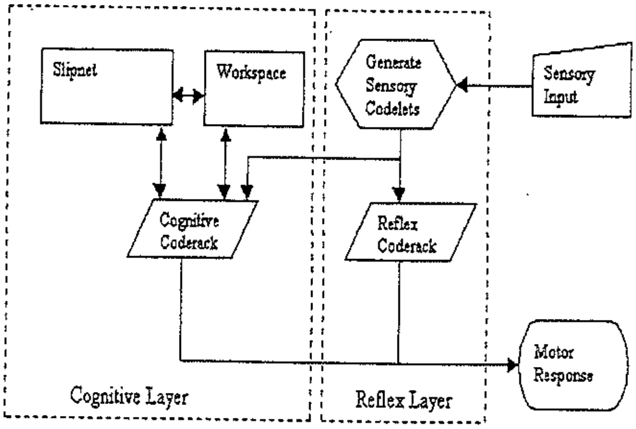

Figure 2: The Madcat Architecture

Figure 2 shows the components of the architecture and their relationships. Simple behaviors are achieved by interactions with a Reflex layer. A priority queue contains instantiations of four basic rules for a given set of readings (called a *snapshot*). These rules are expressed in codelets. The highest value codelet is executed and the queue flushed. This is repeated for each snapshot. The priority queue is a deterministic version of Copycat's coderack where the choice of the next codelet is made probabilistically with a bias toward the higher urgency codelets. This provides the flexibility to discover alternate possibilities.

The remainder of Madcat's abilities derive from the cognitive layer. The cognitive layer contains a probabilistic coderack, the workspace, and the slipnet. The coderack contains codelets whose execution generates robot behavior as well as the emergent structures that serve as a representation of the environment. The workspace serves as the locus of structure-building activity of the codelets from the coderack. Activity in the workspace biases codelet choices in the coderack. The slipnet contains nodes and links that dictate the data to which the codelets respond and the kinds of structures they build. The slipnet topology changes in response to activity in the workspace but its nodes and links remain fixed. The *entropy* reflects how well emergent structures fit into the data the robot encounters and affects the biases of the system. A high entropy inclines the system toward random behavior and perception of different patterns in the data. With low entropy the system gravitates toward the established structures.

The control functions for the robot are made available as C functions that can be linked into developed software. The Madcat architecture itself is implemented in C++. Besides the C-based interface of the robot, the choice of C++ was dictated by the need for real-time behavior. We are building in Java an interface to the architecture that will be used as a development and testing tool.

The Behavior of Madcat

The first goal of the Madcat architecture was to demonstrate that certain basic competencies, roughly those of Brooks (1991a), could be implemented using this emergent architecture. The chosen behaviors are obstacle avoidance, wandering, and wall-following. Obstacle avoidance is defined as the behavior of moving to avoid a collision. Wandering is defined as the behavior of choosing a random direction of motion when no other particular movement is required. We define wall-following as the behavior of moving approximately parallel to the nearest surface, without necessarily moving nearer to that surface to do so.

In the behavior-based approach of Brooks (1991a) these behaviors would be supported by individual interacting layers, each capable of a particular behavior. In an emergent architecture, such as Madcat, a few simple rules interacting among all the data readings give rise to the appropriate behavior. Instead of layers, an emergent architecture relies on competition between peer behaviors to generate coherent global behavior.

There are four basic rules for responding to the data readings. For readings that come from the sonar sensors above either wheel the robot should move forward to follow the surfaces which reflected the signals from those sensors. For readings that come from sonar sensors clockwise from either wheel but not beyond the forward or rear sensors the robot should rotate clockwise to become parallel with the surfaces that reflected the signals from those sensors. Analogous rules hold for readings from sonar sensors counterclockwise from the wheels. If the robot senses contact from one of the six regions of the bump sensor, then it should back up a small amount and turn away from the region to avoid further contact. When each of these rules is given a priority proportional to the proximity of the readings, the desired three behaviors emerge as a result of the moment by moment interactions of the rules, readings, and features of the environment.

Wall-following can be seen in Figure 3 where the robot moves counterclockwise, turning corners to remain on a course parallel to the nearest wall. Obstacle-avoidance is also demonstrated, as the robot turns in response to surfaces detected in its path. Wandering is subordinate to these first two behaviors and so only appears at the end of the path.

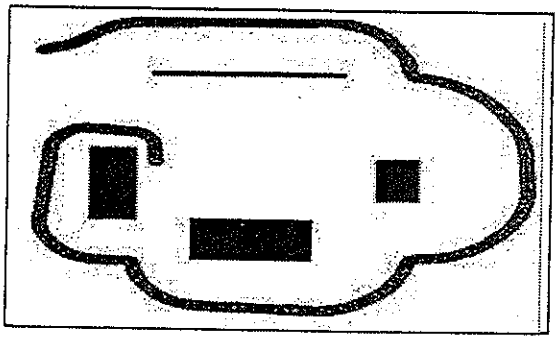

Figure 3: Obstacle Avoidance, Wandering, and Wall-Following

The second goal of the Madcat architecture is to generate emergent structures correlated with environmental features. These support more effective real-time behavior. For example, the direction choice for wandering can be made more useful if the system has a rough model of what it has already encountered. Random directions can be chosen from among those not yet explored. As another example, consider that the sonar sensors produce the same measurement for all readings below 6 inches, preventing the distinction of a corner from a continuation of a nearby wall. If the system contains structures representing a wall located directly ahead, it may use this information to turn away from the wall with which it would otherwise collide.

At the top of Figure 4 the robot passes a convex corner. *Single Surface Element* (*SSE*) structures, corresponding to each of the sonar readings taken while traversing this path, are built in the workspace. Bonds can be built between these SSEs according to the relationships in the data. For instance, *Adjacent Equivalence Bonds* (*AEB*) may be built between SSEs from adjacent sonar sensors if their values are within a certain percentage of each other. *Candidate Surface Bonds* (*CSB*) tend to be built linking a sequence of AEBs, which might possibly constitute a surface. Bonds built from a single snapshot are only tentative. As the data from successive snapshots continue to bear certain relationships, bonds based on those become strengthened. The *Maximum Difference Bond* (*MDB*) identifies the apexes in curved surfaces. These only occur after many snapshots have produced well-established structures. The longer a set of structures is maintained by the data the more likely is its abstraction into an object marker, for example, the convex surface in the bottom of Figure 4.

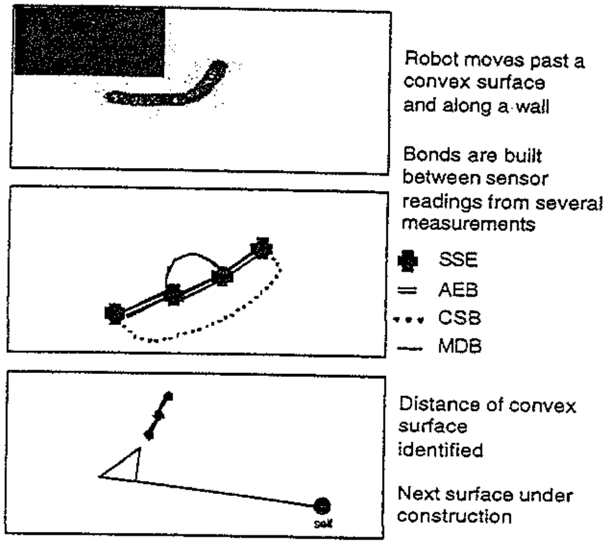

Figure 4: Correlation Between Emergent Structures and Environmental Features

Figure 5 shows the robot approaching a wall to which its sensors are blind. The wall to its left is closer than six inches, below which distance the sonar system is unable to make any distinctions. This makes the approaching wall look like a continuation of the wall to the left. However, during the approach, structures are formed which reflect the sonar readings of the forward wall. If a CSB is built in time, the robot will notice it when scanning its internal surfaces for discrepancies with the environment. At that point it can choose to turn and avoid the wall based on its internal niche model of the world. This demonstrates the use of emergent representation to improve real-time behavior.

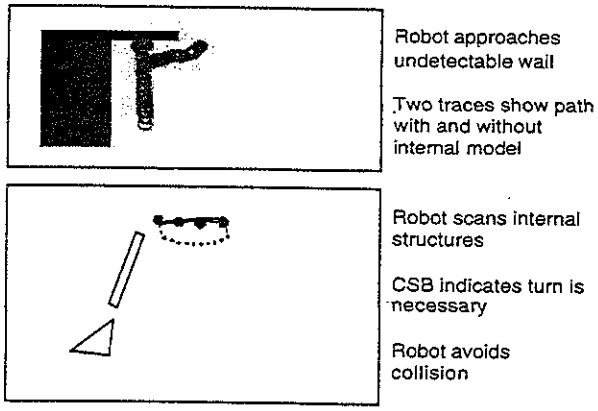

Figure 5: Emergent Structures Aid in Navigation

The role of the slipnet is to provide context-dependence to the competing behaviors in Madcat. For example, consider the creation of an AEB, proposed by some codelet. The comparison of values between adjacent SSEs uses information from the slipnet concerning relative distances of objects in the current environment to discern how precisely the comparison should be made. When the objects detected are at a greater distance from the robot both trigonometric considerations and reliability of the sensors dictate that a greater difference in readings may still correspond to a single surface. Alternatively, when the robot is near its targets, the distinction between surfaces by adjacent sensors is sharper. As another example, the SSEs between which the AEB will be built are themselves chosen probabilistically with a bias coming from the slipnet's indications of which objects have greatest relevance at that moment.

The control architecture also uses *Description* objects to ascertain features of the structures it is building. Description objects may be attached to almost any structure and contain a description for that structure. The method for selecting possible descriptors of an object is dependent on information from the slipnet about the kinds of descriptors that have recently been successfully constructed or used in creating other structures.

Occasionally, the parallel nature of the architecture will give rise to the proposed construction of an object that conflicts in some way with an existing object (e.g., duplication, overlap, opposition). As in the Copycat architecture, the choice of whether to veto the construction or destroy the conflicting object and continue is made probabilistically with a bias that comes from information in the slipnet about which kinds of objects are currently more useful to build. This information comes from the context to which the slipnet has been exposed in the preceding moments of the robot's behavior. Indeed at times the priorities implicit in the current arrangement of the slipnet will bias the probabilistic codelet executions so that the system explores otherwise unnoticeable options.

The entropy measure, much like the temperature in Copycat, is used as a feedback mechanism for the entire architecture. The entropy is derived in part from the slipnet. When entropy of the workspace is calculated, values are obtained from the workspace objects that indicate their relative importance and degree of incorporation into larger structures. The calculation of these values includes the level of activation of the node in the slipnet corresponding to that type of object. So an object whose node in the slipnet has high activation is likely to have greater importance and higher expectation for structure-inclusion. Even the self-organizing feedback in the system is mediated by the context-driven relevance of the concepts in the system. Ultimately information in the slipnet about relative priorities of certain kinds of structures and actions can be used to select or restrict entire classes of behavior.

The slipnet captures this context information through its interactions with the workspace and the codelets. When a codelet successfully builds a structure in the workspace, the slipnet node which originated that codelet gets a boost of activation. That activation spreads to neighboring nodes in the slipnet as a function of the length of the link between them. Thus, related nodes also get some additional activation. As the activation of a node goes up, so does its chances of emitting codelets designed to explore the possibility of building structures in the workspace based on the concept represented by that node.

Activation naturally decays in the slipnet so that over time, if no new objects of a given type are being built, then codelets stop being produced to look for them. Of course there is a certain low probability for generating any type of codelet so the system never stops discovering new possibilities. The mechanisms of the slipnet capture the priorities indicated from the context of recent interaction of the environment and drive the decisions in the entire system.

Further Research

There are two specific areas of further development. The first is to use the internal models of environmental features to augment visual decomposition algorithms used with the color vision camera. The worm algorithm (McGonigle 1998) is commonly used, but it is easily misled. The presence of sonar edges in the internal model can help to corroborate edges found by a variant of the worm algorithm. This kind of synthesis is important in the intelligence of living organisms. We would like to build models that mimic this capacity.

The second extension to our research is related to the idea that events in the environment enable certain behavior sets and disable others. We would like to model the sudden shift of priorities and behaviors in a system in response to events in the environment. Certain colors will be established as triggers for the system. When these are seen, changes in the links in the slipnet and the priorities of codelets occur which override the bias to explore and complete internal models in favor of seeking out a resource or avoiding danger.

Conclusion

We offer both a definition and an instantiation of intelligent problem solving in robotics based in evolving complex adaptive systems. We refine the behavior-based approach to robotics by requiring that representation, redefined as the emergence of structures coupled to the environment through behavior, be given greater focus. We believe that the four issues of embodiment, emergence, symbolic behavior, and representation will be very important in the challenging task of understanding intelligent activity in changing problem domains.

We have demonstrated the feasibility of an emergent architecture in solving simple robotics problems. We have demonstrated that emergent structures in an embodied architecture can be behaviorally correlated to features of the environment, producing niche models useful for generating adaptive behavior. Work is underway using this architecture for improved visual decomposition algorithms and environmentally triggered behavior shifts.

Acknowledgments

This research has been supported at the University of New Mexico by the NSF CISE Research Infrastructural award CDA-9503064 and by the NASA PURSUE Program (PAIR) Grant No. NCC5-350. The contributions of Andy Claiborne, Matthew Fricke, Tim Mitchell, Monica Rogati, and Len Lopes have been invaluable.

References

Brooks, R. (1991a). Intelligence Without Representation. Reprinted in Luger, G. (ed). 1995. *Computation & Intelligence*. 343-364. Cambridge: MIT Press.

Brooks, R. (1991b). New Approaches to Robotics. *Science* 253:1227-1232.

Brooks, R. and Stein, L. (1994). Building Brains for Bodies. In *Autonomous Robots* 1: 7-25. Boston: Kluwer Academic Publishers.

Clark, A. (1997). *Being There*. Cambridge: Bradford Books/MIT Press.

Holland, J. (1998). *Emergence: From Chaos to Order*. Reading, MA: Perseus Books.

Holland, J. (1986). Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. Reprinted in Luger, G. (ed). 1995. *Computation & Intelligence*. 275-304. Cambridge: MIT Press.

Luger, G. (1994). *Cognitive Science: The Science of Intelligent Systems*. San Diego: Academic Press.

Luger, G. and Stubblefield, W. (1998). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. London Addison Wesley.

Marshall, J. (1999). *Metacat: A Self-Watching Cognitive Architecture for Analogy-Making and High-Level Perception*. PhD Dissertation, Indiana University. Bloomington, IN.

Maturana, H. and Varela, F. (1980). *Autopoeisis and Cognition*. Dordrecht, Holland:D. Reidel.

McGonigle, B. (1998). Autonomy in the making: Getting robots to control themselves. In *International Symposium on Autonomous Agents*. Lanzarote: Oxford University Press.

Mitchell, M. (1993). *Analogy-Making as Perception*. Cambridge: Bradford Books/MIT Press.

Steels, L. (1996). The origins of intelligence. In *Proceedings of the Carlo Erba Foundation Meeting on Artificial Life*. Berlin: Springer-Verlag.

Steels, L. (1995). Intelligence - Dynamics and Representations. In *The Biology and Technology of Intelligent Autonomous Agents*. Berlin: Springer-Verlag.