Emergent Representation in a Robot Control Architecture

Andy Claiborne, Matthew Fricke, Len Lopes,
Joseph Lewis and George Luger
{carcare, mfricke, lenlopes, jalewis, luger}
@cs.unm.edu
Department of Computer Science; FEC 323 University
of New Mexico
Albuquerque, NM 87131
February 8, 2000

Abstract

We offer a definition of representation based in dynamical systems. Then we present Madcat, a robotic control architecture that uses emergent representation to develop an internal model coupled to the environment through its behavior. The design is inspired by the Copycat program (Mitchell 1993). We present details of the Madcat architecture, the interface and development tool, and some results. A robot control architecture with these features will behave more robustly and autonomously in changing environments. Further research on vision and motivation will rely on the same design.

Introduction

We present a cognitive model adhering to the constructivist insight that representation occurs as a product of the active construction of experience from perception. This model exists as a control program for a robot whose task is to generate a model of its environment which abstracts sequences of sensory data into perceived objects, allowing the robot to behave more robustly than it could without the object model. Robots with greater flexibility, adaptibility, and robustness are possible using such an architecture. These would be useful in situations where a robot must autonomously adapt to its changing surroundings, for instance in unmanned space landings.

Our work builds on research from several disciplines. These include: work on behavior-based robotics (Brooks and Stein 1994); work on the dynamical nature of representation and intelligence (Steels 1995 and 1996); work on the organization of living systems and their coupling with an environment (Maturana and Varela 1980, Clark 1997); and work on fluid representation in the unique architectures of the Copycat family from Mitchell and Hofstadter (Mitchell 1993).

The active synthesis of representations from a perceptual stream derived from embodiment in the environment creates the experience of objects. This active construction results in (sometimes temporarily) persistent structures in the mind--the knowledge-embodying model. Traditionally, it is these structures, rather than the processes of which they are a product, whose nature has been the focus of the cognitive tradition. However, the peculiar fluid quality of these structures has resisted explanation. A shift of focus in recent years, generated in part from the study of complex adaptive systems, has motivated research into the dynamic processes underlying these structures. Models built on the insight that representation is implicit in behavior and dynamics have begun to appear. Such models also provide a test-bed for the expectation of embodied cognition research that representations only have meaning in the context of embedded experience.

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 real-world environments. This concern motivated a shift in thinking about the organization of intelligence.

The subsumption architecture (Brooks 1991a) marked the beginning of *behavior-based robotics*. Behavior-based robotics emphasizes layers which produce behaviors directly from input rather than 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 abilities to react to dynamic features of an unpredictable environment and to generate robust behavior despite sensor uncertainty are signatures of the 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 intelligence. However, too little emphasis has been placed on the role of representation. Much research remains to be done on the role of emergent structure in a dynamical model of representation. This new conception of representation should accompany the new framework for robotics. We believe that embodiment is so important in this endeavor that building robots is a necessity.

Fluid Representation and Copycat

Our research builds upon Copycat, one of the first computer programs to model the dynamical processes from which symbolic or representational behavior emerges. Copycat solves letter-string analogy problems, for instance if "abc" becomes "abd" what does "ijk" become? Such seemingly simple analogies involve processes that are at the core of intelligent behavior. Copycat's most crucial feature is the *slipnet*. The slipnet is a semantic network with spreading activation and multiple kinds of links among its nodes, some of which are variable in length. The dynamical processes which develop representational structures also 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, 'opposite' links become shorter and more likely to be traversed, and further processes to explore 'opposite' are generated. 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 data.

The Madcat Architecture

Although the Copycat architecture has great merit as a starting point for a general model of embodied intelligence, one of its limitations is its singular point of interaction with its environment (the letter-string analogy problem). The Madcat project explores how a similar architecture can be used to produce the experience of persistent features of the environment through an ongoing dialog with sensory data from that environment. The emergence of structures coupled to the environment through behavior is a defining feature of intelligence, which we call *behaviorally coupled representation*. We believe a dynamical system like Copycat will support behaviorally coupled representation. The behavior-based robotics paradigm serves as a beginning framework for the model.

To create an embodied architecture capable of ongoing interaction with a dynamic environment we embed an architecture similar to Copycat in the control systems of a robot. The robot we use 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 combination of the ideas from Copycat and the Nomad robot produced the project name *Madcat*. The ultimate goal is a robot which, from its emergent exploratory behavior, builds a flexible representation of its environment to improve its real-time behavior.

The architecture will be implemented in two main stages which are responsible for the reflex behavior and the cognitive behavior of the robot. By cognitive behavior we mean the use of the emerging world model to guide the robot's choices. 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 using Java to build an interface to the architecture that will be used as a development and testing tool.

Reflex Behavior

The behavior of the first stage is analogous to a biological system's reflex actions. It allows the robot to receive information and respond to its environment, but does not incorporate any representations of its environment. Our goals were to get familiar with the robot, to establish that the architecture is sufficiently fast to interact with its environment in real time, to get familiar with writing emergent software, and to develop base-level behavior that can be used by more complex cognitive machinery later. In particular, we want the robot to follow walls and avoid obstacles.

Our robot has 6 bump sensors and 16 sonar sensors around its perimeter. Most of the robot's behavior will be determined by its sonar. We can imagine the sonar sensors as being divided into 6 sections by the long lines in Figure 1. Each pair of lines frames a sonar sensor; the rectangles are the robot's wheels.

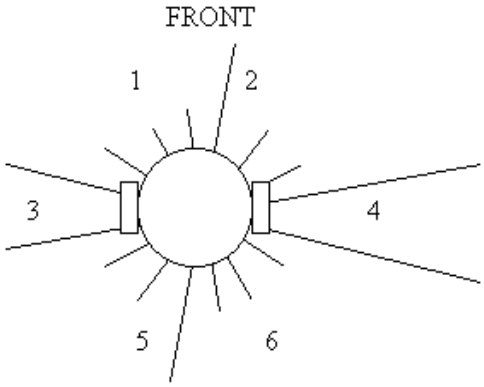

Figure 1: Sonar Sensor Arrangement for Madcat

If the closest sonar reading comes from the sensors in section 3 or section 4, then the robot is positioned parallel to a wall and all it has to do is move forward. If the closest sonar reading is in section 1 or section 6 then the robot has to turn right to move into a position to follow a wall. If the closest sonar reading is in section 2 or 5 then the robot has to turn left to move into position to follow a wall. If the robot bumps into something, it should rotate away from the object. When the robot follows these simple rules, it will also avoid obstacles. The decision to make the front and rear sensors turn the robot right or left is arbitrary. The structures supporting the later cognitive behavior are reflected in the implementation of this algorithm.

At the top level, our code is essentially an infinite loop. At each iteration, each bump sensor and sonar sensor generates a *codelet* that contains instructions to move the robot in a particular way. Each sonar codelet is assigned an urgency that is proportional to the inverse square of the sonar reading, and each bump codelet is assigned the highest urgency. When the codelets are created, they are inserted into a container called the *coderack*. The code then picks the codelet with the highest urgency from the coderack and executes it. The contents of the coderack are then deleted, and it is filled with a new set of codelets.

Cognitive Behavior

We want to give the robot the ability to explore unknown reigons and build internal represtenations of them. In order to do this we need more diverse codelets, a more complex coderack, the slipnet, and the *workspace*.

Each codelet is responsible for a small piece of behavior. Some codelets tell the robot to move in certain ways, other codelets refine the adaptive behavior of the program, and still other codelets modify its internal representation. The global behavior of the robot emerges from the interaction of many competing codelets working in parallel on various structure-building efforts.

In order to effectively pick the codelets that seem most important, we store them in the coderack. The coderack automatically stores each codelet in one of seven bins, based on their importance. Codelets are chosen probabalistically from the coderack, biased by importance, and then executed.

When a codelet is created, we determine its importance based on the slipnet. The slipnet in Madcat is a network of concepts related to the physical world, such as 'forward', 'reverse', 'right', 'left', 'near', and 'far'. Each concept is activated by representational structures in the workspace that relate to features of the physical world. For example, if the robot has a lot of internal structure that suggests close objects, then the concept for 'near' in the slipnet would get a lot of activation. This activation spreads throughout related nodes. With our example, when the activation is poured into 'near', all of the concepts directly tied to 'near' would get a piece of the action. When a new codelet is built, it checks how relevant its function is to the current state of the slipnet. If the codelet corresponds to a concept in the slipnet that currently has a lot activation, then the codelet is inserted into the coderack with a very high importance. Conversely, a codelet that is related to a concept in the slipnet with low activation, then it will be given a very low importance when it is inserted into the coderack.

The structures that influence the slipnet reside in the workspace. The workspace can be visualized as an area where codelets build structures related to the physical world. Initally, when the workspace has no pre-existing structures, pressure from the slipnet leads to a very high preference for codelets that build on the raw sensor readings. So, the codelets usually build many structures that correspond to a single sonar reading. When these get built, they pour activation into concepts in the slipnet that motivate looking for patterns in the data. This activation in turn tends to give high importance to codelets that look for patterns in the data. The objects built by these codelets identify interesting features of the data. When these objects are built, they activate concepts in the slipnet that relate to physical features of the world. These newly activated concepts give high importance to codelets that are capable of building structures that abstract the emerging structures in the workspace. This feedback loop continues to build higher-level objects until there is high coherence among the components of the program. Entropy is the measure of coherence of the structures in the program; it is high when the objects have little coherence and falls as the coherence among the components increases. At that point the program has gleaned as much information as possible from the current set of sonar readings.

Although this is a good model for examining a single set of sonar readings, we want the robot to be able to travel through its world and examine it from many different vantage points. To solve this problem, we designed our workspace to contain a set of *snapshots*, each of which functions like the workspace described in the previous paragraph. We also designed new types of codelets that are capable of building structures that identify patterns that span multiple snapshots. For example, if a codelet identifies a pattern that has existed for several consecutive snapshots, it will build a structure that flags the pattern as a useful abstraction relating to the outside world. The robot keeps track of these abstractions and how they relate to one another, and they comprise the robot's internal representation of its external world.

The Interface and Development Tool

Evaluating the robot's behavior in its environment is essential to understanding this emergent model. However, observing the results of our work acting as whole, while instructive, cannot give us the low level information we need to tune, refine, and evaluate our work. We need a way to look "under the hood". Visualization using a graphical user interface (GUI) is a great way to explore and tune the cognitive engine of our robot. The GUI we have developed to represent the internal processes of the engine has to be flexible enough to allow different views, and sometimes entirely different paradigms of representation. These views must complement each other and provide intuitive imagery. Madcat is information rich and in every aspect of designing the GUI we have been careful to allow the user to easily eliminate and simplify the displays. This makes it easier to focus on the aspect of interest. At the same time we felt the user should be able to view the system in as much detail as desired. Our goal in creating this GUI is to balance these two desires while providing the user as much control over every aspect of the GUI as possible.

Most of the interesting information about the Madcat engine is in the slipnet and workspace representations since these are the two areas from which the robot's decisions emerge. This is where the bulk of our design effort has been spent. The basic visual design of the slipnet display has been implemented. Since the slipnet's components are fixed, a simple node and link graph is appropriate. The activation levels of the nodes and their associated links are represented by coloration. The higher the activation level the shorter the wavelength of the color.

Clicking on a node or link displays additional information. This allows layering of information so that the user is only a click away from further details about individual elements of the slipnet without being overwhelmed by extraneous information.

The workspace display provides more of a design challenge since its principle role is to display structures being created in response to the changing environment. This requires a display that can represent a dynamically changing system while many relationships are created and destroyed at runtime. The wealth of information and complexity in the workspace necessitates a multi-windows approach. The workspace exists not just in two dimensions, as with the slipnet, but in three. This third dimension is composed of the snapshots the robot uses to make inferences about the temporal aspect of its environment. Rather than writing a true three-dimensional model we represent the workspace as a series of two-dimensional representations. This allows the user to quickly see the state of the system at any one time without having to decode it from a complex three-dimensional whole. Several snapshots can be viewed side by side.

The entropy is represented using a simple histogram which graphs the entropy levels over time. This allows the user to quickly identify patters in the the overall coherence of the system. Lessons learned from Copycat argue that the entropy is a very useful indicator of the robot's operation. The coderack can be represented simply with counters of the various codelet types.

In addition to monitoring the real-time actions of the robot as it engages its environment, the GUI will be able to replay the internal states of the system from a file recorded by the robot. This will allow us to slow down the action in order to better understand the system's properties and to troubleshoot problems. The advantages of this ability to change the speed of playback should not be underestimated since it allows us to focus, step-by-step, on specific behaviors in much the same way the different views allow us to zoom-in on specific structures.

In this study we hope to gain some understanding of emergent behavior. We need to be able to zero-in on both the microcosm of codelets and bonds, as well as on the large-scale structures and behaviors they are designed to produce. This is analogous to our understanding of neurology. We cannot understand brain function in organisms by looking at synapses or neurotransmitter molecules at one end of the scale without also understanding the structure of the medulla oblongata or cerebrum. The GUI we are developing will allow us the flexibility and granularity to view and interpret the Madcat system at many levels of detail, including the identification of higher level functioning.

Results

The first goal is to demonstrate that certain basic competences could be implemented using this emergent architecture. The chosen behaviors are obstacle avoidance and wall-following. Obstacle avoidance is defined as the behavior of moving to avoid a collision. 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 these would be supported by interacting layers, each capable of a particular behavior. In an emergent architecture a few simple rules interacting among all the data readings give rise to the appropriate behavior. Instead of layers, this architecture relies on competition between peer behaviors to generate global behavior.

Wall-following can be seen in Figure 2 where the robot turns corners to remain parallel to the nearest wall. Obstacle-avoidance is also demonstrated, as the robot turns in response to surfaces detected in its path.

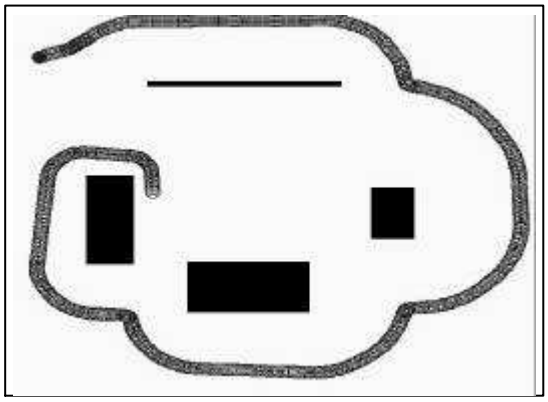

Figure 2: Obstacle Avoidance, Wandering, and Wall-Following

The second goal is to generate emergent structures correlated with environmental features. These afford more effective real-time behavior. For example, the direction choice for exploration 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 cannot detect differences between measurements for readings below 6 inches, preventing the reflex behavior from distinguishing a corner from a continuation of a nearby wall. If the system contains emergent structures representing a forwardly located wall, it may use this information to turn away from the wall with which it would otherwise collide.

Shown in Figure 3 are the emergent structures built by the robot as it passes a convex corner. *Single Surface Element* (*SSE*) structures, corresponding to each of the sonar readings taken while traverseing 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 could constitute a surface. Bonds built within a single snapshot are only tentative. As the data from successive snapshots continues to bear certain relationships, bonds based on those become strengthened. The *Maximum Difference Bond* (*MDB*) identifies 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 it will be abstracted into an object marker, such as the convex surface in the bottom of Figure 3.

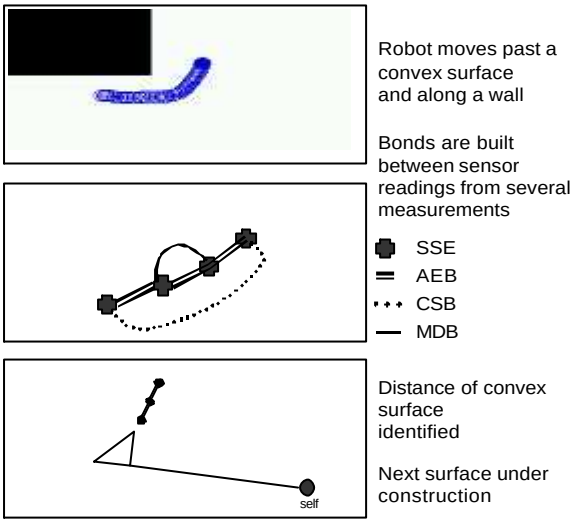

Figure 3: Correlation Between Emergent Structures and Environmental Features

Figure 4 shows the robot approaching a wall which its sensors cannot detect. The wall to its left is closer than six inches, below which the sonar makes no distinction. This makes the approaching wall look like a continuation of the wall to the left. However, during the approach, structures form that 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 model of the world. This demonstrates the use of emergent representation to improve behavior.

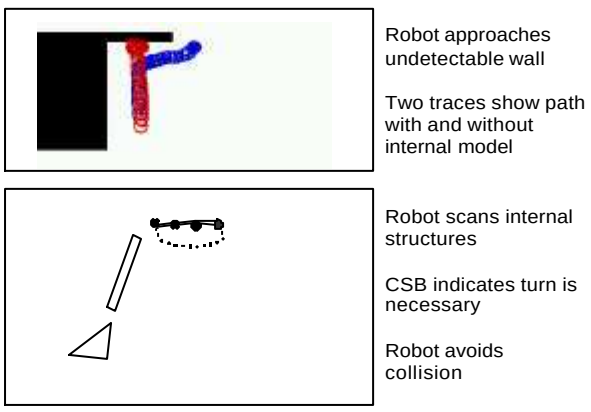

Figure 4: Emergent Structures Aid in Navigation

Further Research

There are two specific areas of further development with which we are already engaged. The first is to use the internal models to augment visual decomposition algorithms using the color vision camera. Edge-detection is a hard problem. Established algorithms have some success but are easily misled. The presence of sonar edges in the internal model can help to corroborate edges found by variants of these algorithms. This kind of synthesis is important to the intelligence of living organisms. We would like to build models that mimic this capacity.

The second of these goals 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 act as triggers for the system. When these are detected by the camera, 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 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 embodiment, emergence, and representation are important for understanding intelligence. 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 correlated through behavior to features of the environment, producing useful models for generating adaptive behavior. Work is underway using this architecture for improved visual decomposition algorithms and environmentally triggered behavior shifts.

Acknowledgment

This research has been supported at the University of New Mexico in part by the NASA PURSUE Program (PAIR) Grant No. NCC5-350 award PP-52-99SU funded by the NASA MURED Division and by the NSF CISE Research Infrastructural award CDA-9503064. The contributions of Tim Mitchell and Monica Rogati 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.

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

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.