

# Computational Adaptive Autonomy: A Generalization of the Copycat Architecture

**Joseph A. Lewis**  
Department of Computer Science  
San Diego State University  
San Diego, CA 92182  
(v): 619-594-2014  
(f): 619-594-6746

**Jamie R. Lawson**  
Lockheed-Martin Orincon Defense  
4770 Eastgate Mall  
San Diego, CA 92121  
(v): 858-795-1426  
(f): 858-452-0271

{[lewis@cs.sdsu.edu](mailto:lewis@cs.sdsu.edu), [jamie.lawson@lmco.com](mailto:jamie.lawson@lmco.com) }

**Keywords:** Computational perception, autonomy, adaptation, AI

*Abstract*—We describe a generalization of Copycat, an important computational architecture for high-level perception. We show how this generalization simplifies the application of computational perception to problems previously beyond the reach of the fluid analogy-making principles underlying Copycat. We also discuss the ways in which this generalization addresses the fundamental problem of brittleness, and embraces a new standard for autonomy and adaptation in artificial systems.

## 1.0 Introduction

Copycat [1,2] provided a model of the fluid rearrangements of concepts essential for the construction of rich analogies, applied this model to the problem of making letter string analogies, and showed the importance of analogy making in high-level perception. The Copycat system solves only “toy” problems, but its importance lies in its emergent model of fluid concepts, its novel approach to knowledge representation, and in the insight it provides for dealing with brittleness. For these merits, John Holland has described Copycat as one of the most important artificial intelligence experiments extant [3,4]. Yet certain features of the Copycat architecture, and its rather Byzantine original implementation have made it difficult to use for solving other problems.

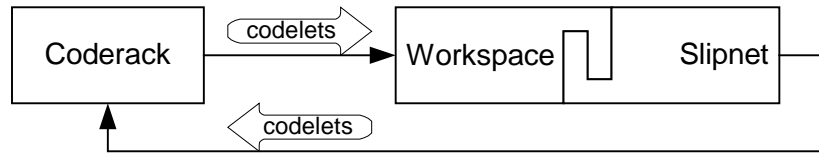
The work here deals with some of these shortcomings and discusses a generalization of the Copycat architecture which supports applications that have:

- The ability to make good decisions based on incomplete or ambiguous information.
- The ability to adapt both their behavior and their objectives based on both a priori knowledge and on knowledge gained from experience.
- Flexible interfaces to the external environment and to the designer during the process of specifying and instantiating the framework for a particular domain.
- A more general kind of autonomy than has been realized in past information systems.

### 1.1 Introduction to the Copycat and Madcat Architectures

The Copycat architecture includes three components: the coderack, the workspace, and the slipnet. Messages from the workspace to the slipnet are handled ad hoc by tight coupling between the two

components. Agents known as codelets carry instructions between the coderack and the workspace, and also between the slipnet and the coderack.



**Figure 1: Copycat Components**

Codelets are small programs that perform work on perceptual structures managed by the workspace. While awaiting their turn to execute in the workspace, codelets are stored in the coderack, which is essentially a stochastic priority queue. When a codelet's number is called, the coderack sends it to the workspace. The workspace may reject the codelet, or execute its instructions. Codelet execution can cause new workspace structures to be created or existing structures to be modified or even removed. These topological changes in the workspace are communicated to the slipnet—another perceptual structure. The slipnet expresses the current tendencies to action by sending new codelets to the coderack.

The slipnet uses a graph to represent tension and resistance between concepts, under pressure provided by a stimulus that consists of three letter strings: the first is a string before some transformation is applied, the second is the same string after the transformation is applied, for instance “‘abc’ goes to ‘cdf’”. Copycat's goal is to discover and represent the transformation, and then apply it to the third letter string. Pressure also comes from the system in that the system's future behavior depends on what the system is already doing.

This last quality was elaborated by Lewis in the Madcat program [5,6]. Madcat applied this unique architecture to the problem of mapping and navigation in a mobile robot. The robot's interactions and behavior in its environment offered a source of ongoing, time-varying embodiment. This demonstrated the generality of Copycat's approach, and its potential for solving real-world problems. Because the robot navigation problem included a temporal dimension absent in the Copycat setting, Lewis added a fourth component, the mapnet, providing a memory in which recurrent patterns can be elevated to the level of concepts. The slipnet allows the system to be informed by what it is already doing, and the mapnet allows the system to be informed by what it has done recently. Madcat's innovations amplified the importance of emergent representation that is dynamic and coupled closely with behavior in an environment.

Madcat's engagement with its environment required the introduction of a partial sufficient knowledge representation allowing the system to produce useful behavior in the kinds of ambiguous situations often found in the real world but often leading to brittleness in artificial systems. Some practitioners [7] believe that the observed brittleness in artificial systems results from the limitations of current knowledge representations. Others [8] believe that the problem lies in the reliance on representation itself, rather than the constraints of any particular representation. Increasing evidence suggests that the traditionally assumed primality of knowledge representation is misguided, but that knowledge representation nonetheless plays a role in intelligent behavior. Partial sufficient representation is one interpretation of the role that knowledge representation should play.

Partial sufficient representation results from the hypothesis that knowledge representation is not what we once thought it was. It does not capture concepts, nor does it, at the other extreme, simply get in the way. Representation is what is leftover once concepts have emerged. It drives the system along the way, allowing the system to be affected by what it is already doing. This effectively turns the brittleness problem around, driving representation with behavior instead of driving behavior with representation. It also speaks to Brooks' admonitions about representation being the wrong (or at least too costly) abstraction: we simply represent as much as necessary to be able to produce effective behavior, and then we move on.

## **2.0 New Design Choices for the Generalization of the Architecture**

The designers of both Copycat and Madcat recognized how the basic architecture could be generalized. While such changes were outside the scope of those efforts, this current work acts on some of their

recommendations, and provides further refinement not previously considered. These innovations are motivated by observations evident even in the coarse discussion above:

- For practical purposes, the coderack, workspace, and slipnet are just data structures. The state of these data structures is regulated by activity from the other components, but the drivers can be easily decoupled from the data structures.
- Two of the three links between these data structures are facilitated by codelets. The third link is peculiar but does not need to be so.
- There is no magic number of components. Different applications may call for different components.

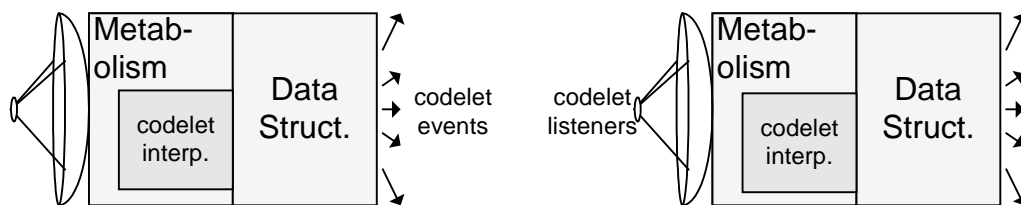
The work presented here exploits these observations to provide a more flexible architecture for high-level computational perception. This architecture is called Starcat (the Kleene closure of the -cat architectures) [9].

The tight coupling between Copycat's workspace and its slipnet is one of the difficulties that has made it daunting to apply the architecture to problems besides letter string analogies. We believe that the simple elegance of the Copycat vision can also be captured in the software and in doing so we expect it to become much easier to build systems that utilize the peculiar power of fluid analogy-making.

One of the first goals of this current work was to decouple the workspace and slipnet components through event-oriented programming so that different versions of a workspace or slipnet could be introduced without rewriting the other component. The slipnet then becomes an observer of the workspace, and responds to asynchronous events that occur in the workspace. It ceases to be concerned with the inner workings of the workspace, and so new versions of either component can be easily introduced.

It quickly became evident that the other components behaved in a similar way. The workspace could be treated as an observer of the coderack, and the coderack as an observer of the slipnet. Further, in each case, codelet events could carry the necessary instructions to drive the system. As a result, coderacks, workspaces, and slipnets are black boxes in Starcat. Each can listen for codelet events and each produces codelet events that other components can listen to. So not only can a new workspace be introduced, but multiple workspaces can listen to the same coderack, and completely new kinds of components (like mapnets) can be introduced without changes to or extensions of the foundation of the architecture.

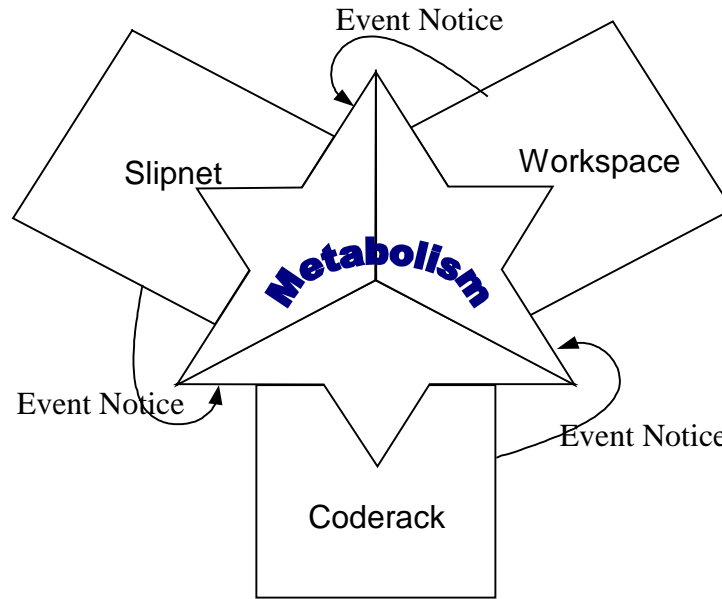
The observation that the main Copycat components were basically data structures led to a more clean separation of the data structures from the controller code that drives them. As a result, each component has a data structure class, and a "metabolism" to control the data structure.



**Figure 2: Event-driven Refactoring of Copycat Components**

The reuse patterns for metabolisms and data structures are different. The same coderack data structure may be mated to many different coderack metabolisms in different problem settings. Testing is a special kind of setting, and is greatly simplified because a metabolism with well understood and proven behavior can be used to drive a new or poorly understood data structure, and vice versa.

The metabolism is the object that listens for the codelet event and then passes it to a codelet interpreter for processing. The metabolism's interpretation of the codelet instructions determines modifications to the data structure. The codelets regulate, but do not specify the dynamics of the component data structure. The metabolism, on the other hand, interprets the codelets and specifies the dynamics of the data structure.



**Figure 3: Copycat Program Described as Starcat Components**

Starcat's object-oriented implementation also affords it expressiveness that would have been difficult or impossible to achieve in Copycat. However many of these innovations are beyond the scope of this paper.

### 3.0 New Principles Underlying Starcat

The observations that made possible the progression of changes in the Starcat design described above derived from an a priori commitment to emerging principles of complex adaptive systems and autonomous open-ended behavior. Simultaneously, those changes in the design have strengthened those commitments in the architecture. Here we discuss the two most important of these commitments.

#### 3.1 A New Framework for Autonomous Behavior

Copycat confronted brittleness by showing how fluid concepts and analogy-making support the ability to make choices with incomplete or ambiguous information. While no letter string analogy is decidedly "right", some analogies are supported by more evidence than others, providing confidence that we have an acceptable solution and can move on to the next problem. Madcat's partial sufficient representation exploited this feature of Copycat's dynamics to produce autonomous behavior in a real environment. The same idea is at the heart of autopoiesis, an important biological theory of autonomous behavior [10]: there are many reasonable responses to a given set of stimuli. So an intelligent creature's behavior may be triggered, but not specified by its environment. This idea, amplified in the Starcat architecture, opens the door to a new model of computational autonomy.

Traditional definitions of autonomy used in information science involve the ability of a system to operate without human intervention, or to determine its behavior from its own experience. Perhaps because of the implicit assumption that information systems must perform useful work, most of these theories ignore what may be the quintessential feature of autonomous systems seen in nature: they use their experience to determine not just their behavior, but also their objectives! With Starcat we develop a new definition of autonomy that overcomes this deficiency, introducing a framework for meta-goals to confront two obvious problems that result from including self-defined goals as a requirement for autonomous systems. These problems are: how to make a truly autonomous system settle on goals that lead to useful work and how to program a system that defines its own objectives. By determining its own goals, Starcat can avoid many fundamental causes of brittleness.

### 3.2 Starcat as a Complex Adaptive System

Starcat employs principles from complex adaptive systems (CAS) to soften the impact of deleterious behavior. In essence, complex adaptive systems can allow unproductive or counter productive behaviors to occur because no single behavior is significant enough to damage the system.

Complex adaptive systems are a means to achieving emergence, and they provide models for capturing some of the subtlety of biological systems. Importantly, the “complexity” of complex adaptive systems lies in their behavior and not in the rules that govern them (not unlike the complexity that unfolds from very simple rules in chaotic dynamical systems). Typical examples of complex systems, like Hofstadter’s ant colonies or artificial immune systems, use very simple agents but many of them. The agents interact through trivial protocols, with limited information available to any particular agent. Complex aggregate behavior results because each of the smaller components pushes on its neighbors, and in their multitudes the agents push their local effects through the system producing complex, coordinated global behavior.

Starcat exploits CAS in several ways. The first is in the tradeoff between exploration and exploitation. Starcat’s codelets—small behavioral primitives that might be thought of as micro-behaviors—execute in a continual avalanche that manifests the system’s overall activity. No one codelet has any significant impact on the state or global direction of the system. Different behaviors can be tried. If their effect is negative, then the probability of trying similar behaviors in the near future is reduced, but the damage from trying a behavior is negligible and quickly corrected.

The second way Starcat exploits CAS deals with the dynamic management of partial sufficient representation. Codelets operate on chunks of the partial sufficient representation. As codelets run, they build up or break down structural elements of a knowledge representation that is vaguely like a semantic network. Special codelets called “surf” codelets are continually at work washing away pieces of the representation that no longer serve a useful purpose under existing conditions. Other “builder” codelets assemble representational structures, or co-opt existing parts of the representation to be used in a slightly different way.

Since the conditions faced by the system undoubtedly change, a codelet that has a negative impact now might have a positive impact at some other point in time. Since multitudes of codelets are run, and any type of codelet has some probability of running, a type of codelet that was useless some time ago but is needed now will get to run, and if it produces a positive effect on the system now, more codelets of that type will have opportunity in the immediate future. As a result, the system adapts. The flow of codelets changes to produce the behavior that is needed at the present time.

## 4.0 Future Work

The Starcat architecture and framework are best suited to problem domains with certain characteristics. Starcat is a good candidate when:

- Working on the problem means producing behavior rather than producing answers or solving problems.
- The problem space itself changes as the system works on it.
- Coping with overwhelming yet incomplete data is paramount.
- Producing behavior in the domain requires recognizing and acting upon conceptual similarities between different sets of stimuli.

One proof-of-concept application has already been built—a simulation of an ant-colony algorithm for finding shortest paths. The ant-colony shortest-path algorithm is first and foremost engaged in producing the behavior of the ants. Finding an “answer” emerges as a side-effect of that activity. Also, the space itself is different with each successive step of the algorithm (e.g., through pheromone deposits and ant interactions). Such an algorithm rarely explores the complete “data” of the problem, nor does it have the opportunity to. And, crucially, the rules of the agents are simple and must be applied in many different yet loosely similar situations.

The main lessons learned from these initial ant colony experiments deal with the character of emergent representation and the relationship between representation and stigmergy. These results are discussed in more detail in [11].

Experiments will continue with ant-colony and swarm algorithms using the Starcat framework, as will related experimentation on the evolution of component design and development of the “instantiation interface”. An early project, demonstrating that Starcat is at least as powerful as its predecessors, will instantiate systems that reproduce the behavior of both Copycat and Madcat. Development is underway now on a set of standard pluggable components to facilitate experimentation with the architecture and its application to a variety of domains. The code will be made available at <http://www.starcat.org> during the first quarter of 2004. Other domains of interest include evidentiary link discovery, the construction of agent societies where each agent is an instance of the Starcat architecture, and the use of the architecture to deal with image classification [12,13].

Codelet evolution is an area of active and ongoing Starcat research. Biological systems adapt on many different timescales. Most notably, they evolve over long time scales to changes in their environment, and they innovate with whatever resources they have on short to medium time scales. The panda provides a classic example. The panda evolved as a carnivore over geological time scales. But fairly recently its prey disappeared and it had adapt as an herbivore. Herbivores have opposable thumbs to peel their food while carnivores use their fifth claw to help bring down prey. The panda has no thumb, and so it innovates effectively with its extended wrist bone. Codelet activity works vaguely like this. The behaviors that can emerge are those that can be constructed from the available types of codelets. But Starcat can’t currently “grow a thumb” if the set of codelet types is insufficient. So one current Starcat project is attempting to apply genetic programming to the codelet population in order to evolve new behavioral primitives when they are needed to address novel features in the environment.

## 5.0 Conclusions

The emergence of structure or pattern evolved through coupling with an environment is a defining feature of intelligence. Steels [14] says that such intelligence can be realized through a general-purpose dynamical architecture. *Starcat* is our attempt to build this architecture. Such an architecture will need to capture emergent properties of interactive behavior, and form concepts about and representations of the environment. These are the defining features of Starcat’s heritage, to which we add further requirements of autonomy, adaptation and open-endedness. To the basic premises of analogy-making as perception we have, in Starcat’s design, added the notion of partial, sufficient representation and insights about aggregate behavior in complex adaptive systems. The AntCat, the simulated shortest-path ant colony algorithm, is the first successful instantiation of the architecture. Future applications and experiments are being designed to demonstrate the generality and utility of Starcat’s mechanisms.

## 6.0 Acknowledgments

Thanks to Melanie Mitchell for reifying Hofstadter’s vision and for guidance in developing Madcat. Several current graduate students have made useful contributions to the Starcat effort, including Adnan Adil, Rafael Ambrosi-Clavijo, Dimitri Guala, Louis Hung, Thomas Nguyen, and Minnie Thomas. Additionally, BBN Technologies provided support in the ongoing quest for institutional support.

## 7.0 References

- [1] Hofstadter, D., *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books, 1995.
- [2] Mitchell, M., *Analogy-Making as Perception*. MIT Press, 1993.
- [3] Holland, J., *Emergence: From Chaos to Order*. Reading, MA: Perseus Books, 1998.

- [4] Holland, J., "Genetic Algorithms as an Engine for the Study of Complex Adaptive Systems", keynote talk, *Genetic and Evolutionary Computation Conference (GECCO 03)*, Chicago, IL, 2003.
- [5] Lewis, J., *Adaptive Representation in a Behavior-Based Robot: An Extension of the Copycat Architecture*, Ph.D. Dissertation, University of New Mexico, Albuquerque, NM, 2001.
- [6] Lewis, J. and Luger, G., "A constructivist model of robot perception and performance", *Proceedings of the 22<sup>nd</sup> Annual Conference of the Cognitive Science Society*. Philadelphia, PA, 2000.
- [7] Lenat, D., "CYC: A Large-Scale Investment in Knowledge Infrastructure", *Communications of the ACM*, Vol. 38, Number 11, November 1995.
- [8] Brooks, R. (1991). "Intelligence Without Representation", reprinted in Luger, G. (ed). *Computation & Intelligence*. 343-364. Cambridge: MIT Press, 1995.
- [9] Lewis, J. and Lawson, J., "Starcat: An Architecture for Autonomous Adaptive Behavior". *Proceedings of the First Annual Hawaii International Conference on Computer Sciences*, January 15-18, 2004. Honolulu, HI, 2004.
- [10] Maturana, H. and Varela, F., *Autopoiesis and Cognition*. Dordrecht, Holland: D. Reidel, 1980.
- [11] Lawson, J. and Lewis, J. "Representation Emerges from Coupled Behavior". To appear in *Workshop Proceedings of the Genetic and Evolutionary Computation Conference 2004*. Seattle, WA. June 24-26, 2004.
- [12] Bolland, S. & Wiles, J., "Adapting Copycat to Visual Object Recognition", *Proceedings of the Tenth Australian Conference on Neural Networks (ACNN'99)*, 1999.
- [13] Mitchell, M., *Perception and Analogy-Making in Complex Adaptive Systems*. <http://www.cse.ogi.edu/~mm/analogy-vision.html>, 2003.
- [14] Steels, L., "The origins of intelligence", in *Proceedings of the Carlo Erba Foundation Meeting on Artificial Life*. Berlin: Springer-Verlag, 1996.