Jrl Syst Sci & Complexity (2006) 19: 1–8

STUDYING COMPLEX ADAPTIVE SYSTEMS

John H. Holland

Received: 15 November 2005

Abstract Complex adaptive systems (cas) – systems that involve many components that adapt or learn as they interact – are at the heart of important contemporary problems. The study of cas poses unique challenges: Some of our most powerful mathematical tools, particularly methods involving fixed points, attractors, and the like, are of limited help in understanding the development of cas. This paper suggests ways to modify research methods and tools, with an emphasis on the role of computer-based models, to increase our understanding of cas.

Key words Agent-based systems, classifier systems, complex adaptive systems, computer-based models, credit assignment, genetic algorithms, parallelism, rule discovery, signal-passing, tags.

Many difficult contemporary problems center on complex adaptive systems (cas). Cas are systems that have a large numbers of components, often called agents, that interact and adapt or learn. A short list cas problems points up their pervasiveness:

- Encouraging innovation in dynamic economies.
- Providing for sustainable human growth.
- Predicting changes in global trade.
- Understanding markets.
- Preserving ecosystems.
- Controlling the Internet (e.g. controlling viruses and spam).
- Strengthening the immune system.

Despite substantial differences in detail, cas share four major features:

(i) Parallelism. Cas consist of large numbers of agents that interact by sending and receiving signals. Moreover, the agents interact simultaneously, producing large numbers of simultaneous signals.

Biological cells, for example, typically use proteins as signals. These proteins operate in reaction cascades and cycles, providing positive and negative feedback to other cascades and cycles. The interactions of these proteins must be tightly coordinated if the cell is to continue to function.

(ii) Conditional action. The actions of agents in a cas usually depend on the signals they receive. That is, the agents have an IF/THEN structure: IF [signal vector x is present] THEN [execute act y]. The act may itself be a signal, allowing quite complicated feedbacks, or the act may be an overt action in the agent's environment.

John H. Holland
Department of Electrical Engineering and Computer Science, College of Engineering, University of Michigan, 1301 *Beal Avenue, Ann Arbor, Michigan* 48109-2122, *USA*. Email: jholland@umich.edu.

Interlocking sequences of signal-processing rules become programs that are executed in parallel, with all that implies for flexibility and breadth of repertoire.

(iii) Modularity. In an agent, groups of rules often combine to act as "subroutines". For example, the agent can react to the current situation by executing a sequence of rules. These "subroutines" act as building blocks that can be combined to handle novel situations, rather than trying to anticipate each possible situation with a distinct rule. Because potentially useful building blocks are tested frequently, in a wide range of situations, their usefulness is rapidly confirmed or disconfirmed.

In biological cells, for example, the citric acid (Krebs) cycle consists of 8 proteins that interact to form a loop. The citric acid cycle is a basic component of all aerobic organisms, ranging from bacteria to elephants.

(iv) Adaptation and evolution. The agents in a cas change over time. These changes are usually adaptations that improve performance, rather than random variations. Adaptation requires the solution of two problems: the credit assignment problem and the rule discovery problem.

The credit assignment problem arises because overt information about performance (payoff, reward, reinforcement, or the like) is often irregular and partial. That is, an agent's performance is the result of an intricate skein of interactions extending over space and time. It is rare that there is information that overtly picks out "stage-setting" options that lead to later improvements in performance.

The play of a game of strategy, such as chess or Go, provides a useful metaphor. After a long sequence of moves, the player receives notification of a "win" or a "loss" and, perhaps, an indication of the size of the win or loss. There is little information about which "stage-setting" moves along the way were critical to that performance. The problem, then, is to determine which of these stage-setting moves might be useful in future games. Similarly, in a biological cell, the "reward" of reproduction results from the interactions of hundreds to thousands of signaling proteins over hours, days, or years.

The rule discovery problem arises when it becomes obvious that some of the agent's rules are ineffective or detrimental. Replacing ineffective rules with randomly generated new rules will not do. That would be much like inserting instructions at random in a computer program. The object is to produce new rules that are plausible in terms of the agent's experience.

There is a mitigating factor that is helpful in resolving the rule discovery problem. Though cas exhibit perpetual novelty, there are repeating sub-patterns, much as in a game of chess. In chess, these sub-patterns have names like "fork", "pin", "gambit", and so on. In cas these sub-patterns constitute building blocks that can be exploited. Said another way, the critical step toward generating plausible rules is discovering building blocks that have served in good rules already in use. This is an important topic that I will discuss later in the paper.

1 What Are the Techniques for Studying Cas?

The first step in studying cas is to note the similarities in several relevant formal approaches.

Control theory	Economics	Biological cells	Games
process variables	activities	phenotypic features	board configurations
operating costs	activity costs	metabolic costs	board evaluation
objective function	profit	fitness	payoff
control policy	plan	reaction net	strategy

The first row gives the way each formalism represents the state of the system. The second row gives the cost associated with each action in the system. The third row gives the value assigned to predetermined "goals", and the last row names the technique(s) used to guide the system to goals.

These similarities immediately suggest that there is value in taking a cross-disciplinary approach to the study of cas. Each formalism is the result of many years of careful study, and each has its own strengths and weaknesses. Comparing corresponding parts of each formalism, we can sort out the parts relevant to cas studies. By so-doing we can take advantage of these long histories, avoiding dead ends already explored.

One common feature of all these formalisms is a reliance on the differential calculus, especially partial differential equations (PDEs). PDEs, however, have limitations as tools for studying cas. The conditional (IF/THEN) action of agents means that an agent's behavior is not easily approximated by the linear techniques that give PDEs their power. It is much as if one were trying to describe a computer program by PDEs.

The difficulties are exacerbated because the agents' rules are continually changing. Agents rarely reach equilibrium, and the combination of conditional action, regular innovation, and perpetual novelty, disrupts the formation of attractors. Optima and steady states are, at best, short-lived in such a context. Add to this that agents are continually revising and updating their information, much as with the regular updates used in weather prediction. As a result chaotic effects are only occasionally influential.

In short, the most powerful theorems associated with PDEs are only occasionally relevant. Similar considerations apply to standard statistical techniques such as regression and Bayesian nets. The traditonal technique of reduction – study the parts, then add up the parts' behaviors to get the behavior of the whole – does not work. The interactions as well as the parts must be studied.

With this in mind we turn to another technique for modeling cas: exploratory computer-based models. Exploratory computer-based models have much in common with the traditonal thought experiments (gedanken experimenten) of physics. One selects some interesting mechanisms and then explores the consequences that occur when these mechanisms interact in some carefully contrived setting. These settings are often not achievable in a laboratory, hence the "laboratory" resides in the head.

The programmed computer greatly enhances the possibilities of thought experiments. Using a computer, the exploratory model can be described with great rigor and executed without mental bias. It is worth noting, in passing, that the definition and execution of computer-executable model is even more rigorous than a mathematical proof. In mathematical proofs we use conventions and shortcuts so that we and do not have carry out each deductive step. If we did not do so, even the simplest proofs would be intolerably long[1]. However, each step in a computer program must be explicitly specified, else the program will not execute as intended. Of course, computer-executable models usually define special cases, so they lie part way between mathematical theory and laboratory experiments.

As in the case of thought experiments, exploratory computer-executable models define possibilities not actualities. They help us build up our intuition for the mechanisms and interactions the program defines. This is of critical importance in studying cas because the conditional interactions play such an important role. Even simple mechanisms with simple interactions can generate complex behavior. Consider again chess or Go: Fewer than a dozen rules – the counterpart of defining the mechanisms – generate a system with perpetual novelty. After hundreds of years of study we still find new patterns and strategies for playing these games. For a cas the rules that define the agents' behavior are the equivalent of the rules of the game.

2 Defining Agents

To study cas formally, we must provide a way of precisely defining the component agents and their interactions. Each agent in this model is specified by a set of conditional (IF/THEN), signal-passing rules. The IF part of the rule "looks for" certain signals; if these signals are present, the THEN part specifies a signal to be sent.

More carefully, the IF part consists of a set of conditions where each condition specifies a class of signals; if any signal of the specified class is present the condition is said to be satisfied. If all of the rule's conditions are satisfied, then the rule is said to be satisfied. Many rules can be satisfied simultaneously, so they compete to send the signals specified by their THEN part. Moreover, more than one rule can win the competition, so several rules can send signals simultaneously. There is no danger of internal conflict between rules, because additional active rules simply broadcast additional signals. This "parallelism" is an important feature: An agent can compose its actions from a combination of rules, rather than requiring a specific rule for each possible situation The formal specification of a signal-passing system of this kind is called a classifier system (cfs)[2].

The signals sent are collected on a signal list. Formally, this signal list is the agent's internal state. An agent's repertoire is completely determined by its rules, and its actions at any given time are determined by its list of signals. A signal stays on the signal list only one time-step. To keep a signal on the list it must be repeatedly posted by a winning classifier. That is, as with a television image, the list must be "refreshed" each time step.

Though most of the signals on the signal list are internally generated, some signals on the list originate from the environment via a set of detectors. Some signals on the list can activate effectors causing external actions, such as agent movement. Thus, both the agent's internal "strategizing", and its interactions with other agents, are mediated by signals. A typical cfs-defined agent, then, has five principle components:

(i) A list of classifiers.
This list may be modified in various ways as the agent adapts to its environment.

(ii) A list signals.
This list changes each time step, in accord with the output of the classifiers that win the competition.

(iii) A set of detectors.
Detectors code information about the environment into signals.

(iv) A set of effectors.
Effectors have conditions, like the classifiers, that are satisfied by signals. The action part of an effector causes some change(s) in the environment.

(v) A set of reservoirs.
Reservoirs determine the agent's "needs" (equivalent of food, shelter, and the like). Certain effector actions, at appropriate places in the environment, cause the reservoirs to be filled. In a typical model the reservoirs are depleted at a constant rate. When a reservoir is close to empty a "low reservoir signal" continues to appear on signal list. The efficiency with which the agent manages to fill its reservoirs gives a measure of performance. Fitness is thus implicitly defined.

3 Formal Specification of a Cfs

A computer-based model of a cfs requires a proper language for representing classifiers. For simplicity of exposition I will use a particularly simple version of a classifier system, though there are many varieties[3]. In this version, all signals are strings of length k, where each position

contains one letter from the two-letter alphabet $\{1, 0\}$. For example, if $k = 5$, a typical signal would be 11010. The set of all possible signals is the set $A = \{1, 0\}^k$. In a similar fashion, the set of all possible conditions, $C = \{1, 0, \#\}^k$, is the set of all strings of length k over the alphabet $\{1, 0, \#\}$. A classifier rule has either one condition, $c \in C$, or two conditions, $c \in C$ and $c' \in C$. The THEN part of the classifier is a signal $a \in A$. A rule is written as c/a (one-condition rule) or $c, c'/a$ (two-condition rule). I will concentrate on two-condition rules, assuming the obvious simplifications for one-condition rules.

[It is easy to provide for signals of variable length and rules with multiple conditions, but it does complicate the exposition. Interestingly, the restricted system described here is still computationally-complete.]

A condition $c \in C$ is satisfied by a signal $m \in M$ if

(i) at each position in c that has a 1, the signal has a 1 at that position;

(ii) at each position in c that has a 0, the signal has a 0 at that position.

(iii) at positions where c has a #, no requirement is made on the signal (that is, the condition "doesn't care" what occurs at positions where it has a #).

For example, with $k = 5$, the signal 10011 satisfies the conditions 1#### and 10#11, but it does not satisfy either the condition 0#### or the condition 10111.

3.1 Rule Strength (Credit Assignment)

Each classifier rule is assigned a strength that is designed to indicate the rule's past contributions to the overall performance of the cfs.

Strengths in a classifier system are modified in two ways. When an effector causes input to a reservoir, classifiers that have posted signals activating that effector are strengthened (classical conditioning). All other strength changes in a classifier system are the outcome of an ongoing competition that treats the whole classifier system as a kind of marketplace. Each classifier in the system is treated as a go-between (middleman, broker) in this market. At any given time, the "suppliers" of a classifier are those classifiers that posted signals satisfying its conditions. When classifier wins a competition it is the "consumer" of each classifier that has just posted a signal satisfying its conditions.

In more detail, a satisfied classifier makes a bid based on its strength, treating its strength as "cash-in-hand". The highest bidders win the competition, and must pay their bids to their suppliers. For this payment, each winner wins the right to post its signal on the signal list. For the rule's strength to increase, it must have consumers that will let it recoup the payment it made to its supplier. In short, the rule must make a "profit". This procedure is called a bucket brigade algorithm (see [4] for details). It has the great advantage that rules do not have to do any back-tracking – the transactions are all local.

3.2 Rule Discovery

In these models, rules serve as tentative hypotheses about the agent's environment, including hypotheses about other agents in that environment. As a hypothesis, a rule is either progressively confirmed or disconfirmed by subsequent events. That is, confirmed rules are progressively strengthened in their ability to compete with other rules, while disconfirmed rules are progressively weakened.

The object of the rule discovery, then, is to produce new hypotheses (rules) to replace hypotheses that have been disconfirmed. The new rules must be plausible hypotheses in terms of previous experience. In this quest, randomly generating new rules is wildly implausible approach. It has no more chance of success than inserting new instructions at random in a computer program.

One direct way to generate plausible new rules is to employ a genetic algorithm[5]. A genetic algorithm treats strong rules as parents, in effect crossbreeding them. The resulting offspring rules combine characteristics of successful rules, in much that same way that a breeder of horses or corn produces new successful varieties.

3.3 Summary

The basic cycle for this cfs is:

(1) All signals originating from the system's environment (via detectors) are added to the signal list.

(2) All rules check all signals on the signal list to determine which rules are satisfied.

(3) All signals are deleted from signal list.

(4) Satisfied rules enter a strength-based competition; the winning rules post their signals to the signal list.

(5) The bucket brigade algorithm adjusts the strengths of the winning rules.

(6) The genetic algorithm generates new rules to replace weak rules.

(7) Changes in the system's environment, including changes caused by signals activating effectors, are executed.

(8) Return to step (1).

Modularity (e.g., a set of subroutines) is provided when signals acquire tags. A tag is a portion of a signal that satisfies the conditions of several related rules. Modules result when a subset of rules coordinates its activity by the use of a common tag. In this respect, tags serve much like headers on internet messages. Because rules can be modified by the genetic algorithm, tags that give rise to modules can arise as the system adapts[6]. That is, rules of the form IF (signal with apropos tag present) THEN (send signal with that tag) are generated. As in bio-circuits, tags act as motifs that identify, and coordinate, modules. The condition for activation of a module can be made more or less precise through the addition or deletion of a # in the tag region of the conditions of classifier rules in the module.

Rule-based, signal-processing systems of this kind have been tested in a variety of contexts. The interested reader can find detailed discussion of classifier systems in [7].

4 Challenges

It is not likely that precise definitions of "complexity", "emergence", or the like, are necessary for studies of complex systems to proceed rapidly. We still have no sharp definition of "life", "species", or many other critical terms in biology, but the science of biology continues to advance rapidly. However, as in the case of biology, we do need rigorous models that allow us to explore properties that are unique to cas.

Here is a short list of properties that offer new opportunities for understanding and controlling cas:

(i) All cas that have been studied carefully exhibit lever points – points where a simple intervention causes a lasting, directed effect. For example vaccines cause lasting, desirable changes in an immune system, and simple dopants provide high-temperature superconductors. There is no theory that tells us where or how to look for cas lever points.

(ii) All cas have a hierarchical organization of boundaries enclosing boundaries, with signals that are attuned to those boundaries. Without boundaries there cannot be individual histories, and without individual histories selection for fitness is not possible. Thus, Darwinian selection depends upon the origin and development of boundaries. But there is as yet no theory or general model that tells us what mechanisms are sufficient for the formation of boundaries in

an initially homogenous system.

(iii) All cas seem to evolve in an open-ended fashion, wherein an initially simple system exhibits increasing diversity of interaction and signaling. Though there has been an increasing use of genetic algorithms, and other techniques for artificial evolution, there are at present no computer-based models that exhibit open-ended evolution.

There are also some plausible hypotheses that go along with these observations:

(i) Local concentrations of resources, induced by feedback, recycling and boundaries, provide opportunities for the formation of new agents.

(ii) This process of agent formation leads to increasing co-evolution and diversity of agents, with progressively larger amounts of resources "tied up" in agents.

(iii) Under 'tranquil' conditions, increasing agent specialization should be observed.

These pervasive properties and hypotheses suggest several exploratory computer-based models:

(i) Seed machines: Construct models that mimic the development of multi-celled organisms from an initial single cell. Such models would be a natural continuation of von Neumann's epoch-making model of a self-reproducing machine[8].

(ii) Evolving reaction networks: Construct an artificial chemistry that starts from a set of elementary reactions based on tags ("active sites"). The object would be to observe the progressive formation of networks with tag-mediated boundaries and signals (tagged compounds) that interact with those boundaries[9]. Successful models of this kind would provide improved ways of describing ecological networks with feedback, co-evolution, and phenotypic plasticity[10].

(iii) Models of language acquisition and evolution: Construct agent-based models situated in an environment where agent survival depends upon interactions with other agents. The objective would be to observe conditions, if any, where structured, grammar-based language emerges from pre-linguistic cognitive abilities[11].

Finally, at a meta-level, observing cas that regularly produce innovations, e.g. evolving ecosystems, suggests changes in the way we usually conduct research:

(i) Risk-taking. Allow for high failure rates in funding research. Because of the exponential growth of the successes and their "spin-offs", the return from "home-runs" greatly exceeds the losses incurred by the failures.

(ii) Diversity and Parallelism. Follow several paths simultaneously in exploring a given question.

(iii) Credit assignment. Provide ways of rewarding stage-setting activities. The most difficult activity in the conduct of science or business is the early, sometimes costly, activity that makes possible later, obviously good actions. As in the game of chess, it is often an early sacrifice of piece (a gambit) that makes possible a good later move. [Real option theory gives a better approach to these questions than standard decision theory[12]].

What can we expect in the longer run?

(i) "Flight simulators" for organizations. Flight simulators are tested by experts who are not programmers. This kind of model is made possible by an interface, much like that for a videogame, that allows the expert to operate in familiar ways. The expert validates the model, suggesting improvements to the model's programmer.

(ii) A theory of cas that makes possible the principled discovery of lever points. Such a theory should lead to great improvements in everything from drug design to better care for the poor.

(iii) Accounting techniques that allow current, "bottom-line" credit for exploratory research. The key here is a change in the "standard" accounting methods that are used in levying taxes and determining net worth. At present the only important accounting technique that allows

for future effects is the allowance for depreciation. A clearer understanding of cas would extend such current allowances to exploratory research.

The study of cas is a difficult, exciting task. The returns are likely to be proportionate to the difficulty.

References

[1] A. N. Whitehead and B. Russell, *Principia Mathematica*, Cambridge Univ. Press, 1927.
[2] L. Bull and T. Kovacs, *Foundations of Learning Classifier Systems*, Springer, 2005.
[3] P. L. Lanzi, W. Stolzmann and S. W. Wilson, *Learning Classifier Systems*, Springer, 2000.
[4] J. H. Holland, K. J. Holyoak, R. E. Nisbett and P. R. Thagard, *Induction: Processes of Inference, Learning, and Discovery* (2nd Edition), MIT Press, 1989, 71–75.
[5] L. Booker, S. Forrest, M. Mitchell and R. Riolo, *Perspectives on Adapatation in Natural and Artificial Systems*, Oxford Univ. Press, 2005.
[6] J. H. Holland, *Hidden Order: How Adaptation Builds Complexity*, Addison-Wesley, 1995. Chinese translation: Shanghai Scientific & Technological Education Publishing, 2000.
[7] L. Bull, *Applications of Learning Classifier Systems*, Springer, 2004.
[8] J. von Neumann, *Theory of Self-Reproducing Automata*, Univ. Illinois Press, 1966.
[9] J. H. Holland, *Hidden Order: How Adaptation Builds Complexity*, Chapter 3, Addison-Wesley, 1995. Chinese translation, Shanghai Scientific & Technological Education Publishing House, 2000.
[10] F. J. Odling-Smee, K. N. Laland, M. W. Feldman, *Niche Construction: The Neglected Process in Evolution*, Princeton Univ. Press, 2003.
[11] J. H. Holland, Language acquisition as a complex adaptive system, in *Language Acquisition, Change and Emergence* (ed. by J. W. Minett and Wm. S.-Y Wang), City University of Hong Kong Press, 2005, 411–435.
[12] L. Trigeorgis, *Real Options: Managerial Flexibility and Strategy in Resource Allocation*, MIT Press, 1996.