za, and Lev A. Zhivotovsky

lture, or group that is dominating
any words in Old English changed
libacy in the church probably had
tions. There are things that you
sion going on, as Murray said, it
on a lot of other traits that are
know which one of these is going

linguistic changes can work very
nt from the way you say your *a's*
ıcestors of the people who taught
ıdon, which is a different body of
t body that went to Pitcairn, or
nk are really interesting to study,
tical or linguistic differences that
et quantifying.

W. Brian Arthur
Stanford University, Stanford, CA 94305-6084

On the Evolution of Complexity

Abstract: It is often taken for granted that as systems evolve over time they tend to become more complex. But little is understood about the mechanisms that might cause evolution to favor increases in complexity over time. This chapter proposes three means by which complexity tends to grow as systems evolve. In coevolutionary systems it may grow by increases in "species" diversity: under certain circumstances new species may provide further niches that call forth further new species in a steady upward spiral. In single systems it may grow by increases in structural sophistication: the system steadily cumulates increasing numbers of subsystems or subfunctions or subparts to break through performance limitations, or to enhance its range of operation, or to handle exceptional circumstances. Or, it may suddenly increase by "capturing software": the system captures simpler elements and learns to "program" these as "software" to be used to its own ends. Growth in complexity in all three mechanisms is intermittent and epochal. And in the first two is reversible, so that collapses in complexity may occur randomly from time to time.

Illustrative examples are drawn not just from biology, but from economics, adaptive computation, artificial life, and evolutionary game theory.

Complexity: Metaphors, Models, and Reality
Eds. G. Cowan, D. Pines, and D. Meltzer, SFI Studies in the
Sciences of Complexity, Proc. Vol. XIX, Addison-Wesley, 1994

INTRODUCTION

It is a commonly accepted belief—a folk theorem, almost—that as systems evolve over time they tend to become more complex. But what is the evidence for this? Does evolution, in fact, favor increases in complexity and, if so, why? By what mechanisms might evolution increase complexity over time? And can the process go in the other direction, too, so that complexity diminishes from time to time? In this chapter I will discuss these questions and, in particular, three different ways in which evolution tends to increase complexity in general systems.

In the biological literature, there has been considerable debate on the connection between evolution and complexity.[1,11] But much of this discussion has been hampered by the fact that evolutionary innovations typically come in the form of smooth changes or continuous, plastic modifications: in the size of organism,[1] in the morphology of body parts,[13] or in animal behavior,[1] so that increases in "complexity" are difficult both to define and discern. As a result, while most biologists believe that complexity does indeed increase with evolution, and particular mechanisms are often cited, the question remains muddied by problems of definition and observation, so that some biologists have expressed doubts about any linkage between evolution and complexity at all.[11] Fortunately, of late we are beginning to cumulate experience in evolutionary contexts that are not necessarily biological. These contexts include those of competition among technologies and firms in the economy, of self-replicating computer programs, of adaptive computation, of artificial life systems, and of computer-based "ecologies" of competing game strategies. Used as alternatives to biological examples, these have two advantages. Their alterations and innovations are very often discrete and well marked, so that in these contexts we can define and observe increases in complexity more easily. And many are computer based. Thus, they can provide "laboratories" for the real-time measurement and replication of changes in complexity in the course of evolution.

In discussing complexity and evolution in this chapter, I will draw examples from the economy and from several of the other contexts mentioned above, as well as from biology. I will be interested in "complexity" seen simply as complication. Exactly what "complication" means will vary from context to context; but it will become clear, I hope, in the mechanisms as they are discussed. And I will use the term "evolution" often in its phylogenetic sense, as development in a system with a clear lineage of inherited structures that may change over time. Thus, we can talk about the evolution of a language, or of a technology, without having to assume that these necessarily reproduce in a population of languages or technologies.

GROWTH IN COEVOLUTIONARY DIVERSITY

The first mechanism whereby complexity increases as evolution takes place, I will call *growth in coevolutionary diversity.* It applies in systems where the individuals

or entities or species or organisms coexist together in an interacting population, with some forming substrates or niches that allow the existence of others. We may, therefore, think of such coevolving systems as organized into loose hierarchies or "food webs" of dependence, with individuals further down a hierarchy depending for their existence on the existence of more fundamental ones nearer the base of the hierarchy.

When the individuals (and their multiple possibilities in interaction) in such systems create a variety of niches that are not closed off to further newly generated individuals, diversity tends to grow in a self-reinforcing way. New individuals that enter the population may provide new substrates, new niches. This provides new possibilities to be filled or exploited by further new entities. The appearance of these, in turn, may provide further new niches and substrates. And so on. By this means, complexity in the form of greater diversity and a more intricate web of interactions, tends to bootstrap itself upward over time. Growth in coevolutionary diversity may be slow and halting at first, as when the new individuals merely replace uncompetitive, preexisting ones. But over time, with entities providing niches and niches making possible new entities, it may feed upon itself; so that diversity itself provides the fuel for further diversity.

Growth in coevolutionary diversity can be seen in the economy in the way specialized products and processes within the computer industry have proliferated in the last two decades. As modern microprocessors came into existence, they created niches for devices such as memory systems, screen monitors, and bus interfaces that could be connected with them to form useful hardware—computing devices. These, in turn, created a need, or niche, for new operating system software and programming languages, and for software applications. The existence of such hardware and software, in turn, made possible desktop publishing, computer-aided design and manufacturing, electronic mail, shared computer networks, and so on. This created niches for laser printers, engineering-design software and hardware, network servers, modems, and transmission systems. These new devices, in turn, called forth further new microprocessors and system software to drive them. And so, in about two decades, the computer industry has undergone an explosive increase in diversity: from a small number of devices and software to a very large number, as new devices make possible further new devices, and new software products make possible new functions for computers, and these, in turn, call forth further new devices and new software.

Of course, we should not forget that as new computer products and functions for computers appear, they are often replacing something else in the economy. Computer-aided design may eventually replace standard drawing board and T-square design. And so the increase in diversity in one part of a system may be partially offset by loss of diversity elsewhere. Occasionally, in a coevolving system, this replacement of an existing function can cause a *reversal* in the growth of coevolutionary diversity. This happens when the new entity replaces a more fundamental one in the system and the niches dependent on this disappear. In the economy of the last century, for example, there was a steady increase in the numbers of specialized,

interconnected "niche firms" in the horse-drawn transportation industry; so that by the end of the century very many different types of coach builders, harness makers, smithy shops, and horse breeders coexisted. The appearance of the automobile caused all this to collapse, to be replaced, in turn, by a slow-growing network of interconnected niche manufacturers dependent on gasoline technology, oil exploration and refining, and the internal combustion engine. Thus, complexity—diversity in this case—may, indeed, tend to grow in coevolving systems, but it may also fluctuate greatly over time.

Growth in diversity can be observed in several artificial evolution contexts: for example, Tom Ray's Tierra system,[14] John Holland's ECHO system,[6] and Stuart Kauffman's various chemical evolution systems.[7] To take the Tierra example, Ray sets up an artificial world in which computer programs compete for processor time and memory space in a virtual computer. He begins with a single "organism" in the form of a set of self-replicating machine language instructions that can occasionally mutate. This forms a niche or substrate for the appearance of parasitic organisms that use part of its code to replicate—that "feed" on its instructions. Further organisms appear that are immune to the parasites. The parasites in turn form a substrate for hyper parasites that feed on them. Hyper-hyper parasites appear. And so on. New "organisms" continually appear and disappear, in a rich ecosystem of symbiotic and competing machine-language programs that shows a continual net growth of diversity. In several days of running this system, Ray found no endpoint to the growth of diversity. Starting from a single genotype, over 29,000 different self-replicating genotypes in 300 size classes (equivalent to species in this system) accumulated in this coevolving computer ecology.

At this point I want to note several things that apply to this mechanism.

First, the appearance of new entities may, in some cases, depend not so much on the existence of previous entities as on their possibilities in interaction. For example, in the economy, a new technology such as the computer laser printer mentioned above is possible only if lasers, xerography, and computers are previously available as technologies. In these cases, symbiotic clusters of entities—sets of entities whose collective activity or existence is important—may form many of the niches. We could predict that where collective existence is important in forming niches, growth in coevolutionary diversity would be slow at first—with few entities there would be few possibilities in combination and, hence, few niches. But as more single entities enter, we would see a very rapid increase in niche possibilities, as the number of possible niche clusters that can be created undergoes a combinatorial explosion.

Second, collapses will be large if replacement by a new entity happens near the base of the dependency hierarchy; small if near the endpoints. Therefore, the way in which expansion and collapse of diversity actually work themselves out in a coevolutionary system is conditioned heavily on the way dependencies are structured.

Third, two positive feedbacks—circular causalities—are inherent in this mechanism. The generation of new entities may enhance the generation of new entities, simply because there is new "genetic material" in the system available for further

"adaptive radiation." And the appearance of new entities provides niches for the appearance of further, new entities. In turn, these mean that where few new entities are being created, few new entities can appear; thus, few new niches will be created. And so the system will be largely quiescent. And where new entities are appearing rapidly, there will be a rapid increase in new niches, causing further generation of entities and further new niches. The system may then undergo a "Cambrian explosion." Hence, we would expect that such systems might lie dormant in long periods of relative quiescence but burst occasionally into periods of rapid increase in complexity. That is, we would expect them to experience punctuated equilibria.

This mechanism, whereby complexity increases via the generation of new niches, is familiar to most of us who study complex systems. Certainly Stuart Kauffman has written extensively on various examples of self-reinforcing diversity. Yet strangely it is hard to find discussion of it in the traditional biological literature. Bonner's 1988 book, *The Evolution of Complexity*, does not mention it, for example, although it devotes a chapter to a discussion of complexity as diversity. Waddington[16] comes somewhat closer when he suggests that niches become more complex as organismal diversity increases. The more complex niches, he suggests, are then filled by more complex organisms, which in turn increases niche complexity. But he seems to have in mind an upward spiral of internal structural complexity, and not of ecological diversity. An intriguing mention of this mechanism—or something tantalizingly close to it—comes from Darwin's notebooks,[3] p. 422.[1]

> "The enormous number of animals in the world depends, of their varied structure and complexity. . .hence as the forms became complicated, they opened fresh means of adding to their complexity."

But once again this could be read as having to do with internal structural complexity, rather than ecological diversity.

STRUCTURAL DEEPENING

A second mechanism causing complexity to increase over time I will call structural deepening. This applies to single entities—systems, organisms, species, individuals—that evolve against a background that can be regarded as their "environment." Normally, competition exerts strong pressure for such systems to operate at their limits of performance. But they can break out of these limits by adding functions or subsystems that allow them to (a) operate in a wider or more extreme range, (b) sense and react to exceptional circumstances, (c) service other systems so that they operate better, and (d) enhance their reliability. In doing so, they add to their

[1] I am grateful to Dan McShea for pointing out this quotation to me.

"structural depth" or design sophistication. Of course, such functions or subsystems, once added, may operate at their limits of performance. Once again they can break through these limits by adding sub-subsystems according to (a)–(d) above. By this process, over time the original system becomes encrusted with deeper functions and subfunctions. It may improve greatly in its performance and in the range of environment it can operate in. But in doing so, it becomes increasingly complex.

The history of the evolution of technology provides many examples of structural deepening. The original gas-turbine (or jet) aero engine, designed independently by Frank Whittle and Hans von Ohain in the 1930s, for example, was simple.[2] It compressed intake air, ignited fuel in it, released the exploding mixture through a turbine that drove the compressor, and then exhausted the air mass at high velocity to provide thrust. Whittle's original prototype had one moving part, the compressor-turbine combination. But over the years, competitive pressures felt by commercial and military interests led to constant demands for improvement. This forced designers to overcome limits imposed by extreme stresses and temperatures, and to handle exceptional situations, sometimes by using better materials, but more often by adding subsystems.

And so, over time, higher air-compression ratios were achieved by using not one, but an assembly—a system—of many compressors. Efficiency was enhanced by a variable position guide-vane control system that admitted more air at high altitudes and velocities and lowered the possibility of the engine stalling. A bleed-valve control system was added to permit air to be bled from critical points in the compressor when pressures reached certain levels. This also reduced the tendency of the engine to stall. A secondary airflow system was added to cool the red-hot turbine blades and pressurize sump cavities to prevent lubrication leakage. Turbine blades were also cooled by a system that circulated air inside them. To provide additional thrust in military aircombat conditions, afterburner assemblies were added. To handle the possibility of engine fires, sophisticated fire-detection systems were added. To prevent the build up of ice in the intake region, deicing assemblies were added. Specialized fuel systems, lubrication systems, variable exhaust-nozzle systems, and engine-starting systems were added.

But all these required further subsystems, to monitor and control *them* and to enhance their performance when they ran into limitations. These subsystems, in turn, required sub-subsystems to enhance their performance. A modern, aero gas turbine engine is 30 to 50 times more powerful than Whittle's and a great deal more sophisticated. But Whittle's original simple system is now encrusted with subsystem upon subsystem in an enormously complicated array of interconnected modules and parts. Modern jet engines have upwards of 22,000 parts.[2]

And so, in this mechanism, the steady pressure of competition causes complexity to increase as functions and modifications are added to a system to break through limitations, to handle exceptional circumstances, or to adapt to an environment itself more complex. It should be evident to the reader after a little thought

[2] Personal communication from Michael Bailey, General Electric Aircraft Engines.

that this increase of structural sophistication applies not just to technologies, but also to biological organisms, legal systems, tax codes, scientific theories, and even to successive releases of software programs.

One laboratory for observing real-time structural deepening is John Holland's genetic algorithm.[5] In the course of searching through a space of feasible candidate "solutions" using the genetic algorithm, a rough ballpark solution—in Holland's jargon, a coarse schema—appears at first. This may perform only somewhat better than its rivals. But as the search continues, superior solutions begin to appear. These have deeper structures (finer subschemas) that allow them to refine the original solution, handle exceptional situations, or overcome some limitation of the original solution. The eventual solution-formulation (or schemata combination) arrived at may be structurally "deep" and complicated. Reversals in structural depth can be observed in the progress of solutions provided by the genetic algorithm. This happens when a coarse schema that has dominated for some time and has been considerably elaborated upon is replaced by a newly "discovered," improved coarse schema. The hierarchy of subschemas dependent on the original coarse schema then collapses. The search for good solutions now begins to concentrate upon the new schema, which in its turn begins to be elaborated upon. This may happen several times in the course of the algorithmic search.

John Koza's genetic programming algorithm, in which algebraic expressions evolve with the purpose of solving a given mathematical problem, provides a similar laboratory.[8] In Koza's setup, we typically see the algorithmic parse trees that describe the expressions grow more and more branches as increasing "depth" becomes built into the currently best-performing algebraic expression.

In Figure 1 I show the growth of structure as the search for good "solutions" progresses in one of Koza's examples. As we can see, once again this mechanism is not unidirectional. Reversals in structural depth and sophistication occur when new symbolic expressions come along that allow the replacement of ones near the "root base" of the original system. On the whole, depth increases, but with intermittent reversals into relatively simpler structures along the way.

Collapse near the base of a system can be seen in a very different context, the history of science, when new theories suddenly replace old, elaborate ones. An example is the collapse of Ptolemaic astronomy caused by the Kepler-Newton version of the Copernican theory. This novel system, that explained planetary orbits using only a few simple laws, struck at the root base of the hugely complicated Ptolemaic system; and it had such superior explanatory power that the Ptolemaic system never recovered. Similarly, Whittle's jet engine, with its extraordinarily simple propulsion principle, largely replaced the piston aero engine of the 1930s, which had become incurably complicated in attempts to overcome the limitations in operating internal combustion engines at high speed in the very thin air of higher altitudes.[4] And so in evolving systems, bursts of simplicity often cut through growing complexity and establish a new basis upon which complication can again grow. In this back-and-forth dance between complexity and simplicity, complication usually gains a net edge over time.

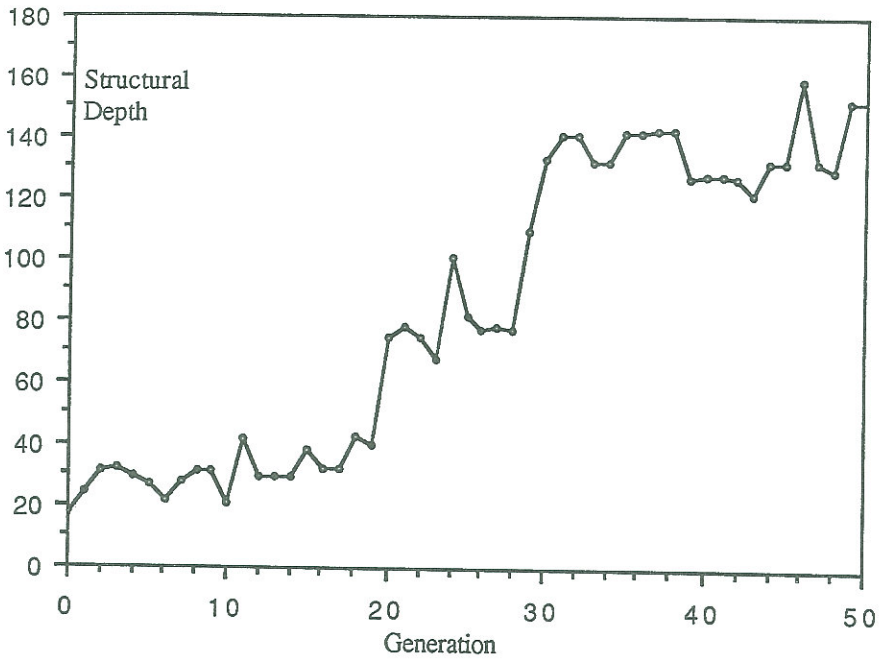
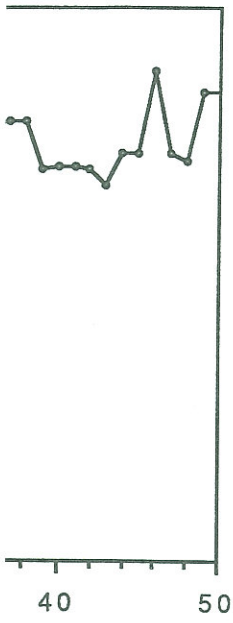


FIGURE 1 Structural depth (number of parts in parse tree) of the currently best expression plotted against number of generations of search in the problem of finding a Fourier series expression to match a given function (from Koza,[8] p. 502).

So far I have described two apparently separate mechanisms. In the first, ecosystems—collections of many individuals—become more complex, more diverse, in the course of evolution; in the second, individuals within ecosystems become more complex, structurally deeper, in the course of evolution. In many systems, of course, these mechanisms operate simultaneously, and they may interact, alternate, and even compete.

This can be seen clearly in Kristian Lindgren's study of strategies that evolve in a game-theoretic setting.[10] Lindgren sets up a computerized model populated by strategies that meet randomly and cumulate profit by playing, one-on-one, a finite version of the iterated prisoners' dilemma. The competing strategies are described as coded bit-strings, where the bits represent memory of previous plays that the strategies can take account of. The strategies can occasionally mutate. Successful ones proliferate in this coevolutionary environment; unsuccessful ones die out. In Lindgren's world, it can clearly be seen that the diversity of strategies increases as new coevolving strategies provide niches that can be exploited by fresh, new strategies, exactly as in the first mechanism I have discussed. But the strategies themselves also become increasing "deep"—their code string or memory lengthens—as



he currently best
he problem of finding
a,[8] p. 502).

chanisms. In the first,
complex, more diverse,
in ecosystems become
n. In many systems, of
nay interact, alternate,

f strategies that evolve
ed model populated by
ng, one-on-one, a finite
trategies are described
revious plays that the
ally mutate. Successful
essful ones die out. In
strategies increases as
d by fresh, new strate-
it the strategies them-
memory lengthens—as

competition rewards increasingly subtle strategies, as in the second mechanism. In fact, the two mechanisms interact in that the arrival of a new, successful, deeper strategy eliminates many of the previous, simpler strategies. Diversity collapses, and with it many of the niches it provides. There follows a phase in which the newer, deeper strategies mutate and proliferate, so that diversity increases again. And so new depth can both destroy old diversity and feed a new round of increased diversity among newer, deeper strategies. In this way, the growth of coevolutionary diversity alternates in a sporadic way with the growth of structural depth in the strategies. This process has obvious parallels in the history of biological evolution. Some biologists suggest, for example, that increased "depth" in the form of the appearance of multicellular, eukaryotic organisms fueled the Cambrian explosion of diversity 600 million years ago.[15]

CAPTURING SOFTWARE

The third mechanism in the growth of complexity that I will propose is completely different from the first two. Actually it has more to do with the rapid emergence of complexity than with any slow growth. It is a phenomenon I will call capturing software. This is the taking over and "tasking" of simpler elements by an outside system for its own (usually informational) purposes. Typically the outside system "discovers" the simpler elements and finds it can use them for some elementary purposes. The elements turn out to have a set of rules that govern how they may be combined and used—an "interactive grammar." This grammar typically allows many combinations of the simple elements; and as the outside system begins to learn this grammar, it also learns to take advantage of the elements in combination. At full fruition, the outside system learns to use this interactive grammar to "program" the simple elements and use them in complicated combinations for its own multipurpose ends.

This mechanism may sound somewhat strange and unfamiliar; so let me clarify it by some examples. A very simple one would be electronics, taken as a technology. As humans, we have learned over the last couple of centuries to "task" electrons to carry out such activities as transmitting sound and vision, controlling sophisticated machinery, and computing. Originally, in the days of Faraday and Franklin, the workings of electrons and of static electricity were poorly understood. And so, uses were few. But in the last century and in the early decades of this one, we began to learn the "grammar" of electricity—the set of operational rules involving induction, capacitance, and impedance that govern the movements of electrons and amplification of their flow. And so we slowly learned to "capture" and "program" electrons for our own use. In this case the simple elements referred to above are electrons. The outside system is ourselves, the human users. The grammar is the laws of electromagnetism. And the programmable outputs are the various technical

uses to which electronics are put. At the output level, there is swift "adaptation." The various technological purposes in which we use electrons as a "programmable software" shift and expand rapidly. But at the grammar and carrier level, in this case, adaptation is absent. The behavior of electricity and of electrons is fixed by physical laws that are, within the human time frame at least, immutable.

Sometimes with capturing software, the interactive grammar is not laid down unalterably, but can itself change and evolve in the process of "capturing" the software. An example is the way in which human language evolved. Early humans learned perhaps several hundred thousand years ago that crude, emitted sounds could be used for communicating warnings, pleasure, or simple needs. Very slowly, and comparatively recently on an evolutionary time scale, they began to generate some elementary rules—a grammar—to organize these into simple concatenated expressions. Eventually, over many thousands of years, these sounds or phonemes plus grammar evolved into a complex interactive system—a language. This could be "programmed" to form statements, queries, and commands that conveyed a high degree of nuance and subtlety.

In this example, the simple, carrier elements are the sounds or phonemes of human speech. The outside system is the human community that "captures" and makes them into a software, a language. And the grammar is the syntactical system that develops to ensure consistency and commonality of meaning. Of course, there is no single, natural syntactical grammar for human language. A grammar must emerge by the slow evolution of a social convention, with constraints exercised by the need for linguistic efficiency and consistency and by the way linguistic activities are organized in the human brain.[9] (Of course, both the human vocal anatomy and brain also changed as a response to the evolution of language.) The overall language that results from this evolutionary process is a programmable software whose potential output we may think of as the set of all meaningful sentences or statements the language can express.

Adaptation in this case can occur at all levels. At the program output level, adaptation is instantaneous. We can think of a sentence uttered as a one-off, extremely rapid adaptation of software output to the purpose of what the sentence is intended to communicate. At the grammar level, adaptation implies change in the language itself. This commonly takes the form of drift, and it happens slowly and continuously. This is because any abrupt alteration or large deviation in grammar would invalidate current "output programs." At the phoneme or simple element level, adaptation—or change and drift—is slowest of all. Slight changes at this carrier level, if not subtle and continuous, might upset all that is expressed in the system. Slow drift may occur, as when vowels shift over the course of a generation or two; but there is a powerful mechanism acting to keep the carrier elements locked-in to a constant way of behaving.

A particularly telling example of capturing software is the way in which sophisticated derivatives have arisen and are used in recent years in financial markets. In this case the outside system is the financial community. It begins by the simple trading of something of value—soybeans, securities, foreign currencies, municipal

bonds, Third World debt, packages of mortgages, Eurodollars—anything to which title can be held. Such items may fluctuate in value and can be swapped and traded. They are called underlyings in financial jargon, and they form the simple, carrier elements of the system I want to consider now.

In early days of such markets, typically an underlying is simply held and traded for its intrinsic value. But over time, a grammar forms. Traders find they can: (a) usefully arrange *options* associated with contingent events that affect the underlying; (b) put several underlyings together to create an associated *index*, as with a stock index; (c) issue *futures* contracts to deliver or obtain an underlying at some time, say, 60 days or one year, in the future; and (d) issue *securities* backed by the underlying. But notice that such "derivatives" as contingent-event options, indices, futures, and securities are themselves elements of value. Thus, they, too, can become *underlyings*, with their own traded values. Once again the market could apply (a), (b), (c), or (d) to these new underlyings. We may then have options on securities, index-futures, options on futures, securities indices, and so on, with trades and swaps of all these.

With such a grammar in place, derivatives experts "program" these elements into a package that provides a desired combination of financing, cash-flow, and risk exposure for clients with highly particular, sophisticated financial needs. Of course, financial markets did not invent such programming all at once. It evolved in several markets semi-independently, as a carrier element was used, simply at first and then in conjunction with the natural grammar of finance.

From the examples I have given, it may seem that the system that uses and captures simple elements to its own uses is always a human one. But, of course, this is not the case. Let me point out two examples in the biological sphere. One is the formation of neural systems. As certain organisms evolved, they began to "task" specialized cells for the simple purposes of sensing and modulating reactions to outside stimuli. These specialized cells, in turn, developed their own interactive grammar; and the overall organism used this to "program" this interconnected neural system to its own purposes. Similarly, the ancestors of the cells found in the immune systems of higher organisms were used originally for simple purposes. Over time, these, too, developed useful rules of interaction—an interactive grammar—thereby eventually becoming a highly programmable system that could protect against outside antigens.

Biological life itself can be thought of in this way. Here the situation is much more complicated than in the previous examples. Biological organisms are built from modules—cells mainly—that in turn are built from relatively small and few (about 50 or so), fairly simple molecules.[12] These molecules are universal across all terrestrial life and are the carriers of biological construction. They are combined into appropriate structures using a grammar consisting of a relatively small number of metabolic chemical pathways. This metabolic grammar, in turn, is modulated or programmed by enzymes. The enzymes doing the programming of course have no conscious purpose. In fact they themselves are the carriers in a second

programmed system. They are governed by a complicated gene-expression "grammar," which switches on or inhibits their production from the genes or DNA that code for them, according to feedback received from the state of the organism they exist in. And so we have one captured software system, the programming of the simple metabolic pathways via proteins or enzymes to form and maintain biological structures, modulated by another captured software system, the programming of proteins or enzymes via nucleic acids and the current state of the organism.

In this case the entire system is closed—there is no outside system programming the biological one to its own purposes. In the short term each organism programs itself according to its current development and current needs. In the long term the overall system—the resulting biospheric pattern of organisms that survive, interact, and coevolve—together with environmental and climatic influences, becomes the programmer, laying down its code in the form of the collection of gene sequences that survive and exist at any time. Of course, without an outside system, we can not say these programmable systems were ever "captured." Instead they emerged and bootstrapped themselves, developing carriers, grammar, and software as they went. Viewed this way, the origin of life is very much the emergence of a software system carried by a physical system—the emergence of a programmable system learning to program itself.

Capturing software in all the cases discussed here is an enormously successful evolutionary strategy. It allows the system to adapt extremely rapidly by merely reprogramming the captured system to form a different output. But because changes in grammars and in carriers would upset existing "programs," we would expect them to be locked in and to change slowly if at all. This explains why a genetic sequence can change easily, but the genetic code can not; why new organisms can appear, but the cell and metabolic chemistry remain relatively fixed; why new financial derivatives are constantly seen, but the securities-and-exchange rules stay relatively constant.[3]

CONCLUSION

In this chapter, I have suggested three ways in which complexity tends to grow as evolution takes place. It may grow by increases in diversity that are self-reinforcing; or by increases in structural sophistication that break through performance limitations; or by systems "capturing" simpler elements and learning to "program" these as "software" to be used to their own ends. Of course, we would not expect such growth in complexity to be steady. On the contrary, in all three mechanisms we

[3] Carriers do change, of course, if they can be substituted for one another easily. For example, options can be built on any underlying; and so, in this case, carriers can and do change rapidly. But the essential property of underlyings—that of being an object that carries uncertain value—remains necessary in all cases and does not change.

would predict it to be intermittent and epochal. And we would not expect it to be unidirectional. The first two mechanisms are certainly reversible, so we would expect collapses in complexity to occur randomly from time to time.

As we study evolution more deeply, we find that biology provides by no means all of the examples of interest. Any system with a lineage of inherited, alterable structures pressured to improve their performance shows evolutionary phenomena. And so, it is likely that increasingly we will find connections between complexity and evolution by drawing examples not just from biology, but from the domains of economics, adaptive computation, artificial life, and game theory. Interestingly, the mechanisms described in this chapter apply to examples in all of these evolutionary settings.

ACKNOWLEDGMENTS

This paper was originally presented at the Santa Fe Institute's Integrative Themes Workshop in July, 1992. I thank Dan McShea, Brian Goodwin, and the workshop participants for useful comments. I am grateful to Harold Morowitz in particular for several conversations on the themes of this essay.

REFERENCES

1. Bonner, J. T. *The Evolution of Complexity.* Princeton: Princeton University Press, 1988.
2. Constant, E. W. *Origins of the Turbojet Revolution.* Baltimore: Johns Hopkins University Press, 1980.
3. Darwin C. *From Charles Darwin's Notebooks*, edited by P. H. Barrett et al., 422. Ithaca: Cornell University Press, 1987.
4. Heron, S. D. *History of the Aircraft Piston Engine.* Detroit: Ethyl Corp., 1961.
5. Holland, J. *Adaptation in Natural and Artificial Systems*, 2nd ed. Cambridge: MIT Press, 1992.
6. Holland, J. "Echoing Emergence: Objectives, Rough Definitions, and Speculation for Echo-Class Models." Mimeograph, University of Michigan, 1993.
7. Kauffman, S. "The Sciences of Complexity and Origins of Order." Working Paper 91-04-021, Santa Fe Institute, 1991.
8. Koza, J. *Genetic Programming.* Cambridge: MIT Press, 1992.
9. Lieberman, P. *The Biology and Evolution of Human Language.* Cambridge: Harvard University Press, 1984.

10. Lindgren, K. "Evolutionary Phenomena in Simple Dynamics." In *Artificial Life II*, edited by C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen. Santa Fe Institute Studies in the Sciences of Complexity, Vol. X, 295–312. Reading, MA: Addison-Wesley, 1991.
11. McShea, D. "Complexity and Evolution: What Everybody Knows." *Bio. & Phil.* **6** (1991): 303–324.
12. Morowitz, H. *Beginnings of Cellular Life.* New Haven: Yale University Press, 1992.
13. Müller, G. B. "Developmental Mechanisms at the Origin of Morphological Novelty: A Side-Effect Hypothesis." In *Evolutionary Innovations*, edited by Matthew Nitecki, 99–130. Chicago: University of Chicago Press, 1990.
14. Ray, T. S. "An Approach to the Synthesis of Life." In *Artificial Life II,* edited by C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen. Santa Fe Institute Studies in the Sciences of Complexity, Vol. X, 371–408. Reading, MA: Addison-Wesley, 1991.
15. Stanley, S. M. "An Ecological Theory for the Sudden Origin of Multicellular Life in the Late Precambrian." *Proc. Nat. Acad. Sci.* **70** (1979): 1486–1489.
16. Waddington, C. H. "Paradigm for an Evolutionary Process." In *Towards a Theoretical Biology*, edited by C. H. Waddington, Vol. 2, 106–128. New York: Aldine, 1969.