Artificial Life's Rich Harvest

Startlingly realistic simulations of organisms, ecosystems, and evolution are unfolding on computer screens as researchers try to recreate the dynamics of living things

Think of it as Biology Meets the Hacker Culture. For 5 days this past June, as some 400 "ALife" enthusiasts descended on Santa Fe's Sweeny Convention Center for the third workshop on Artificial Life, plenary speakers presented artificial ecologies evolving on vivid computer graphics displays and showed startlingly realistic images of plants that computers had "grown" from seed. In the back rooms, graduate students worked into the wee hours making little robots out of Lego blocks and programming them to run through mazes. And on one memorable evening, a group of philosophically minded panelists debated long into the night whether it's possible to create "real" life in a computer—or whether real digital life has been created already, in the form of self-reproducing computer viruses.

But most of all, the workshop participants spent the week of 16 June reveling in an idea: that life can best be understood in terms of computation. After all, goes the ALife argument, life isn't just a collection of structures such as the molecules in a cell or the cells in an organism. Life is also a collection of processes, such as metabolism, or reproduction, or protein formation. And when you look at how these processes unfold through time, they do seem a lot like subroutines being executed in a computer. When you back away far enough from the messy biomolecular details, in fact, they *are* subroutines: little programs encoded on a data tape known as DNA.

Meanwhile, goes the argument, not only is life computation-like, but real computer programs can be disconcertingly lifelike, in the sense that a simple piece of computer code can easily generate behavior that is spontaneous, unpredictable, and astonishingly complex. So why not take this analogy seriously? Why not place the idea of computation at the heart of biology, and try to build new computer models that will give us fundamentally new insights into life?

That was the notion that led Christopher G. Langton of Los Alamos National Laboratory to coin the term "artificial life" and set up the first experimental workshop in 1987. And it's a notion that seems to have increasing appeal: Much to Langton's astonishment, 150 people turned up for that first meeting, and the crowds have been getting bigger ever since. (The second ALife workshop, in 1990, drew some 300 people.) By now, moreover, ALife is no longer just a hackers' playground.

D.R. FOWLER ET AL.

Simple rules, complex result. A green coneflower blossoms in a computer simulation of plant development.

Working biologists are beginning to look at computer simulations for insight into embryonic development, the dynamics of ecosystems, and evolution, among other things. Says Scripps Institute biochemist Gerald Joyce, who has attended all three of the ALife workshops: "That first meeting was like a trip to Radio Shack. But there's been less and less of that each time."

Of course, he adds, not every biologist is quite so sympathetic. "The concern among wet scientists is the limited reality-checking in ALife," he says. "They'll say that simulations are just simulations, whereas real molecules can surprise you—and those tend to be the exciting results, where you get the new insights. It's just that I don't entirely agree: I think that there is something to be learned at the level of first principles from simulations —especially if they suggest new experiments."

Much the same reaction comes from biophysicist Harold Morowitz of George Mason University, editor of the soon-to-be published journal *Complexity* and one of several benchtop researchers to speak at the workshop. After all, he says, there was a time when molecular biology sounded pretty flaky, too. "Asking for physical explanations of life was considered rash hubris," he laughs, thinking back to the late 1940s and his days as a graduate student in Yale University's fledgling biophysics group. "And it's true that a lot of nonsense got said. People were going to derive all biology from Schroedinger's equation. But there was an excitement in those days, a real sense in the air that something new was happening. Well, I had some of the same feeling at the ALife workshop," says Morowitz. "It's not clear yet which ideas are right. But my sense is that computer simulation does allow you to develop radically new approaches to a lot of problems in biology."

A digital greenhouse

At the University of Calgary, to take a particularly vivid example, computer scientist Przemyslaw Prusinkiewicz is modeling growth and development. Development, of course, is the classic illustration of the biology-as-computation analogy: A relatively small number of regulatory genes switch on and off like a series of computer commands, transforming a single, fertilized egg cell into an adult organism with billions of highly specialized cells. So it's no coincidence, said Prusinkiewicz, that the fundamental mechanisms of development can often be abstracted into a simple set of computational rules.

The basic idea was pioneered in the late 1960s by the late Dutch biologist Aristid Lindenmeyer, who was studying the growth of plants. Lindenmeyer pointed out that with just one trivial rule—each time a shoot emerges, say, have it branch into two new shoots—you could describe the development of a hypothetical plant from seedling to complex bush. And by adding just a few more rules, you could likewise model the step-by-step development of branches, leaves, and flowers in three dimensions, as well as the steady spread of hormones and other regulatory compounds from one part of the plant to another. With the advent of advanced graphics workstations, moreover, it is now relatively straightforward to convert the symbolic expressions of such an L- (for Lindenmeyer) system into realistic on-screen images of leaves, flowers, and stems.

"The strength of this approach is that you can model not just the structure of the plant, but the entire development process," says Prusinkiewicz, who worked closely with Lindenmeyer until the latter's death in 1989, and who has continued to develop the L-system approach in collaboration with a number of developmental biologists. The computer is almost literally growing the plant from seed. Obviously, he cautions, "you can't validate a hypothesis with a computer model; that requires going back to the organism itself. But you can show whether or not a hypothesis has a chance of being true."

Recently, for example, he and his collaborators have been applying the L-system idea to study the notion that the basic pattern of development isn't determined by genetics alone; the final form of an organism can also be shaped by the environment, in the form of mechanical crowding and geometric constraints. That's an attractive explanation for, say, the double spiral pattern of the tight-packed seeds in the heart of a sunflower and the geometric pattern of spines on a barrel cactus. And indeed, by allowing the computer to adjust the shape of each part of the plant after each round of applying the developmental rules, he and his co-workers have found that the conjecture works—and in the process, they have produced a flurry of near photographic-quality simulations of sunflowers, daisies, cacti, dandelion seeds, zinnia petals, and even broccoli.

Obviously, said Prusinkiewicz, a top priority in this work is to model morphogenesis in animals as well as plants. But the task is much more complex. "In plant development the cells don't move with respect to one another," he noted in an interview with *Science*. "In animal development, they do." Nerve cells, for example, are notorious for their migrations in the embryo—a behavior that could only be modeled with an exceedingly complicated set of rules, if it could be done at all. As a start, however, he has recently begun working with Yale biologist Leo Buss to devise an L-system that can model gastrulation, the complex folding and migration of cell sheets that first roughs out an animal's body plan.

D.R. FOWLER *ET AL.*
Dueling spines. Crowding influences the geometry of a barrel cactus, as this computer simulation suggests.

Breeding mosquitoes by computer

Prusinkiewicz's work illustrates a recurrent theme in ALife: A complex result—in this case, an intricately formed organism—can emerge from the interaction of many simple elements, such as flower parts or spines. The same approach—of parts interacting according to simple computational rules—can conjure up some of the complexity of a whole ecosystem, as biologist Charles Taylor of the University of California, Los Angeles (UCLA), told the workshop.

Taylor's effort to apply the ALife vision of life as an unfolding program had its roots in the late 1970s, when he was still at the university's agricultural school in Riverside and was trying to build a computer model of the mosquito population in surrounding Orange County. The goal was to test control strategies: How much chemical should you use to kill the mosquito larvae, and when should you spray? Or should you minimize your use of environmentally dangerous chemicals and instead put your efforts into biological controls such as larvae-eating fish?

In principle, Taylor says, building a model to test these questions should have been straightforward. Not only had the county mosquito control authorities collected plenty of data on which to base the simulation—a luxury that is all too rare in the ecosystem modeling game—but the life cycle of mosquitoes is very well known, from the development of the eggs in stagnant water to the female's first bite and back again.

Unfortunately, Taylor says, that first model did not work well at all, and he eventually abandoned it. In retrospect, the problem was that he and his colleagues had followed the conventional approach to simulations in population biology, in which a single variable was supposed to stand for the entire mosquito population, and a complicated set of mathematical equations was supposed to describe the population's rise and fall. But that approach just couldn't capture the complexity of a real ecosystem, recalls Taylor. "In the real world, for example, rain will scour out the breeding sites in gutters and underground storm drains, but it will create new sites in flower pots. Everything interacts. And it was essentially impossible to capture all that with differential equations coded in FORTRAN."

That frustration eventually led Taylor to strike out in a totally different direction: breaking up the population and its environment into a number of simple programs and letting complex behavior emerge from their interactions. The result was RAM, a general-purpose toolkit for ecosystem modeling that Taylor has been developing since the early 1980s in collaboration with UCLA computer scientist David Jefferson and their students. The RAM version of the mosquito model, which he started building several years ago with his student John Fry, includes four different types of breeding sites—swimming pools, storm drains, flood control channels, and gutters—all scattered around the computer's internal map of Orange County in a close approximation of their real distribution, and all programmed to respond in an appropriate way to changes in precipitation and temperature. Meanwhile, the zillions of mosquitoes buzzing through the county are modeled not as individuals, which would have been computationally unthinkable, but as a set of populations: four populations in each region of the county to represent the immature mosquitoes in the four types of breeding sites, and a fifth to represent the adults. Each population contains a set of rules specifying how its survival rate and development time depends on weather and the county's control efforts.

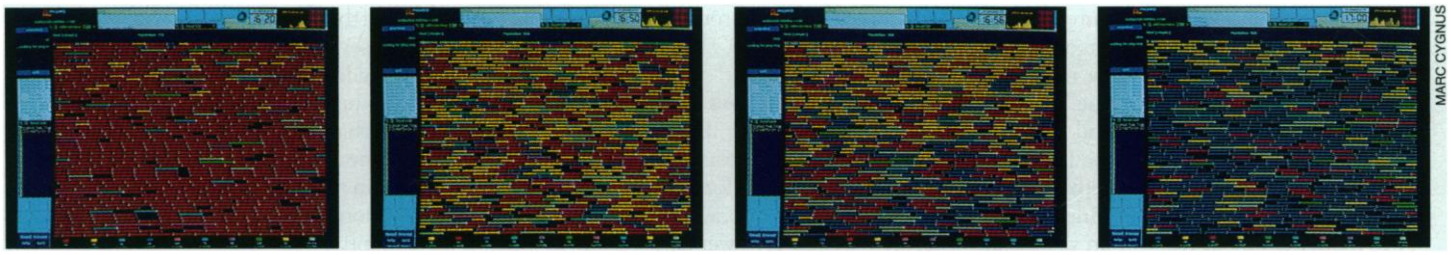
Working closely with Orange County mosquito control officials, Taylor and Fry then fed in nearly a decade's worth of weather records and control protocols. The result: The model reproduced the observed rise and fall of mosquito populations over the past decade quite successfully. And that success has encouraged the UCLA team to turn the model around and try to assess the effectiveness of various control strategies.

At one point, says Taylor, "we found that increasing the use of larvicides had very little effect. In fact, there was relatively little effect when we put larvicide use to zero. At first we thought that there was something wrong with the model. But when we went to the Orange County mosquito control officials, they said, 'Gee, we thought that all the time.' " The model had confirmed their suspicions that the flower pots, old tires, and unused swimming pools of suburban backyards, where larvicides can't be used, shelter more than enough larvae to keep the populations high. Flooding gutters and storm drains with chemicals just doesn't make much of adifference.

Evolution in a cathode-ray tube

In Taylor and Prusinkiewicz's simulations, the simple rules representing the organisms don't change—only their interactions do. But in the grandest process of life, evolution, the rules themselves have to change as new organisms emerge. And that kind of change also lies within reach of the A-life approach, as was evident in one of the most talked-about simulations at the workshop: Tierra, the digital ecosystem created by Thomas Ray of the University of Delaware.

Ray, an evolutionary biologist who has spent much of his career studying the rain forests of Costa Rica, was motivated to develop Tierra (Spanish for Earth) in part by impatience. He wanted to understand how the rich diversity of the rain forest has arisen from the constant competition of organisms. And he knew that he was never going to live



Evolutionary arms race. In Tierra, a model of evolution, an ancestor species *(red)* is overrun by a parasite *(yellow)*, then evolves into a new, immune species *(blue)*. The successor species will soon drive the parasite to extinction.

long enough to see that kind of evolutionary change happening in the field. Computer simulation was an obvious alternative.

But Ray was also guided by his long-standing fascination with the idea of a self-replicating computer program, a notion first explored in the late 1940s by the Hungarian-born mathematician and computer pioneer John von Neumann. Ray found the parallels with biological life irresistible. A self-replicating digital organism would have a kind of genome: the sequence of machine instructions that tell it how to reproduce and interact with its environment. It would have a body that occupied space: the chunk of computer memory where those instructions reside. And it would have a metabolism that consumed a form of energy: the "CPU cycles," or execution steps, that the computer uses to make the instructions actually do something. Furthermore, Ray realized, these same parallels with biology would make the digital world a perfect laboratory for evolution. Just as organic life evolves by natural selection as individuals compete for light, food, and space, digital life would evolve as computational organisms competed for CPU cycles and memory.

Ray started work on his evolving digital organisms in 1989. To create them, he devised a new computer language with precisely 32 different instructions, each of which could be represented by a string of five 1s and 0s. Ray thought of these 32 instructions as analogs of the 64 different 3-base codons of DNA. And they shared with the codons a property that was crucially important if digital evolution was ever going to work: Any mutation—that is, any change from a 1 to a 0 or vice versa—produced another instruction that was still meaningful. (In standard computer languages, a one-bit mutation will almost invariably produce gibberish.)

Once his new language was in hand, Ray used it to write the simplest self-reproducing program he could think of. It had a total of 80 instructions divided into three "genes," or subroutines. When the code was executed, the first gene would measure the digital organism's own length: 80. The second gene would find a chunk of free memory space that was 80 instructions long and reserve it for the creation of a daughter organism. And the third gene would systematically copy all 80 of the parent's instructions into the daughter—occasionally making a 1-bit mistake, or mutation, to make evolution possible. Finally, the daughter organism would be set free as a new, independent organism capable of reproducing itself.

Ray dubbed this 80-instruction beast the "ancestor," and he turned it loose in his computer to grow and evolve in a 60-kilobyte block of free memory that he thought of as the "soup." As expected, the ancestor and its self-reproducing daughters filled up the soup very quickly. The only thing that held them back was mortality, which Ray had instituted in the form of the "reaper": a master program that regularly eliminated the oldest and/or the most error-prone organisms. At the same time, as Ray had also expected, the digital creatures began to evolve and diversify.

"It's not clear yet which ideas are right. But my sense is that computer simulation does allow you to develop radically new approaches to a lot of problems in biology."
–Harold Morowitz

What Ray hadn't expected, however, was that his Tierran ecosystem would evolve so fast. He had resigned himself to putting in years of work before Tierra did anything very interesting. But in the system's first working run, in January 1990, he began to see all manner of ecological phenomena. Once the soup had filled with copies of the length-80 ancestor, for example, there quickly arose a race of length-45 parasites that had lost their copy gene through mutation—but that could commandeer the copy gene of a nearby ancestor to replicate themselves anyhow. The parasites spread through the soup like wildfire, slowing down only when there appeared a fully self-replicating, length-79 mutant of the ancestor that was immune to them. Eventually, in fact, this new creature drove the parasites to extinction—although by then other parasites had already appeared that could breach *its* defenses.

Meanwhile, entirely new reproductive styles were emerging, like that of a race of creatures that Ray dubbed "hyperparasites." Like the ancestor, a hyperparasite was fully capable of reproducing itself. But when attacked by a normal parasite, it would surreptitiously substitute its own code, which the parasite would then continue to copy for the rest of its life. The hyperparasite could thus reproduce itself for free. And so it went. The diversity was extraordinary, and every run was different.

Of course, as Ray himself is the first to point out, Tierra is still a very simple caricature of an ecosystem. But in just the past few weeks, he told *Science*, he has revised the basic Tierra programming language in ways that should enhance the richness and realism of the simulations: by giving the creatures the ability to communicate via digital "hormones"; by giving them "surface tags" that will let them recognize each other; by allowing them to reproduce sexually through the exchange of genetic material; and by allowing the mother organism to regulate precisely what part of its daughters' computer code will execute—thus laying the groundwork for multicellular alliances and differentiation.

All this may yield new insights into evolutionary processes—and it may even feed back into computer science as well. As Ray points out, natural selection gives the Tierran organisms a powerful incentive to be computationally efficient, which is just what human software engineers strive for. In one of his later runs, for example, his length-80 ancestor eventually gave rise to a descendent that could replicate itself with only 22 instructions. So one obvious thing to do, he said, is to require the Tierran creatures to perform some useful computational function before they can reproduce. Over time, mutation and natural selection should give rise to some extraordinarily efficient algorithms for that function—including some that human programmers might not have thought of.

If so, Tierra might be a prime example of how ALife could serve as a two-way street: having brought computational ideas to biology, it might also bring biological ideas to computation.

–M. Mitchell Waldrop