

# Challenges for Biologically-Inspired Computing

Russ Abbott  
Dept. of Computer Science  
California State University, Los Angeles  
and  
The Aerospace Corporation  
El Segundo, California  
0-310-621-3805

[Russ.Abbott@GMail.com](mailto:Russ.Abbott@GMail.com)

## ABSTRACT

We discuss a number of fundamental areas in which biologically inspired computing has so far failed to mirror biological reality. These failures make it difficult for those who study biology (and many other scientific fields) to benefit from biologically inspired computing. These areas reflect aspects of reality that we do not understand well enough to allow us to build adequate models. The failures-to-date are as follows.

1. The apparent impossibility of finding a realistic base level at which to model biological (or most other real-world) phenomena. Although most computer systems are stratified into disjoint and encapsulated levels of abstraction (sometimes known as layered hierarchies), the universe is not.
2. Our inability to characterize on an architectural level the processes that define biological entities in both enough detail and with sufficient abstraction to model them.
3. Our inability to model fitness except in terms of artificially defined functions or artificially defined fitness units. Fitness to an environment is not (a) a measure of an entity's conformance to an ideal, (b) an entity's accumulation of what might be called "fitness points," or even (c) a measure of reproductive success. Fitness to an environment is an entity's ability to acquire and use the resources available in that environment to sustain and perpetuate its life processes.
4. Our inability to build models that allow emergent phenomena to add themselves and their relationships to other phenomena back into our models as first class citizens.

These failures arise out of our inability as yet to fully understand what we mean by emergence.

As an initial step towards surmounting these hurdles, we attempt to clarify what the problems are and to offer a framework in terms of which we believe they may be understood.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*BioGEC'05*, June 25–26, 2005, Washington, D.C, USA.  
Copyright 2005 ACM x-xxxx-xxx-x/xx/xxxx...\$5.00.

## Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence, D.0 [Software]: General, I.6.0 [Simulation and Modeling]: General, J.3 [Life and Medical Sciences], J.4 [Social and Behavioral Sciences], F.1.0 [Computation by Abstract Devices]: General

## General Terms

Algorithms, Design, Economics, Reliability, Experimentation, Languages, Theory.

## Keywords

Emergence, epiphenomena, evolutionary computing, fitness, modeling, process computing, simulation.

## 1. INTRODUCTION

Although there is now a complex network of conferences and journals through which flows a torrent of biologically inspired computing papers, a number of fundamental modeling problems (listed in the abstract) remain unsolved.

The problems pertain to the following interconnected collection of issues. Please note that the following list covers ground which is similar to the list in the abstract, but it isn't exactly the same list, and the issues are not covered in the same order.

1. **What is emergence?** How do we model it? How can we build models in which emergent phenomena, once they emerge, play roles in the model as central and concrete as the phenomena initially built into the model?

Emergence has been the holy grail of biologically inspired computing.

- We understand emergence to be something like a macro phenomena that appears as a by-product of a (generally but not always large) collection of micro phenomena.
- In attempting to understand what we mean by emergence, we (see Bedau [5]) have categorized emergent phenomena as nominal (or benign [17]) (e.g., the emergence of an automobile from its components when they are put together in the right way), weak (the interesting type, e.g., the emergence of foraging effects from the activities of ants, or see Bedau, who uses the glider in the Game of Life as his prototypical example

of weak emergence.), and strong (e.g., vitalism, the emergence of “life” from “lifeless” chemicals<sup>1</sup>).

Also, see Abbott [1] where we define an aggregation to be an (emergent) entity if (a) it has properties that do not apply to its components (e.g., mile-per-gallon for a car) and (b) those properties depend on how the aggregation is held together (again, consider a car).

- Just as we have produced nominally emergent objects and systems for centuries, we have been able to write computer simulations that produce examples of weak emergence (such as “boid” flocks) for two decades. (See Reynolds [16].)

As gliders<sup>2</sup> and boid flocks illustrate, emergent phenomena are often produced and are understood as epiphenomenal by-products of some underlying phenomena, i.e., they are considered to be causally powerless. A flock (as distinct from the boids in it) has no causal role in a boids simulation. A glider (as distinct from the cells that compose it) has no causal role in a Game of Life run.

But as we know, epiphenomenal gliders can be used to do real computation. See Rendell [15] which proves that by using gliders and other Game of Life patterns, the Game of Life can be shown to be Turing complete. And flocks are real entities for those who study migratory behaviors.

So this is a fundamental problem. We don’t know how to produce computational systems in which emergent phenomena (that aren’t explicitly anticipated) can, once they emerge, participate in the system as if they had been built in from the start. Yet that is exactly what happens in the real world: emergent phenomena (and as we shall argue below, we spend most of our lives interacting with emergent phenomenon) are as real and concrete as any other.

2. **What is an entity?** Fundamental and philosophically slippery as this question is, it can’t be avoided. For example, are gliders and flocks entities?

This issue is especially relevant in the case of biological entities. A fundamental difference between biological (and other “higher level” entities) on the one hand and more primitive physical and chemical entities (such as atoms and molecules) on the other is that biological entities require energy from the environment to sustain themselves. Biological entities are what have been referred to as “far

from equilibrium”<sup>3</sup> systems—in contrast with more primitive physical entities, which are at an energy equilibrium.

Unfortunately we do not know how to build computer models in which the fundamental elements are far-from-equilibrium systems. (For a somewhat extended discussion of the distinction between entities that are at equilibrium and those that are far from equilibrium see Abbott [1].)

For an entity to be far-from-equilibrium requires that it is not static, i.e., something is happening to keep it both stable but not in at an equilibrium state. Thus far-from-equilibrium systems are best understood as processes. A hurricane and a river (both of which are discussed below) are non-biological, self-perpetuating, far-from-equilibrium systems. They are not the particular molecules of air and water of which they are composed at any one time. (You *can* step into the same river twice; it just won’t have the same water in it each time.) Thus both are better understood as processes than as things.

Like rivers and hurricanes, biological entities are also self-perpetuating, far from equilibrium processes—but much more complex. When we ask our biologist friends if there is a book something like *Bacteria Maintenance for Dummies: what to do when your pet bacterium breaks* that will tell us how the simplest biological entities perpetuate themselves, we are told there is none.

So here we have a second and a third fundamental problem. What are entities in general, and how can we build useful biological models without having a process architecture framework that describes how biological entities work? Although biology and the social sciences are the most obvious application areas, this is a fundamental problem. We are lacking a general theory of process entities.<sup>4</sup>

3. **How do entities really depend on their environments?** What does fitness really mean?

Most evolutionary computational models depend either on fitness functions or on “health” or “energy” tokens as in computer games. That just won’t do. Nor is fitness either a measure of an entity’s conformance to an idea or, in our opinion, a measure of reproductive success, which is less a measure of fitness than a measure of a correlate of fitness. Fitness to an environment is simply an entity’s ability to acquire and use the resources available in that environment to sustain and perpetuate its life processes.

Process entities (i.e., far-from-equilibrium systems) extract and use energy and resources from their environment to run the processes through which they perpetuate themselves. Fitness is the ability to do this. Yet until we are able to build

---

<sup>1</sup> Were strong emergence, such as vitalism, to be established as a real phenomenon, it would be considered spooky and mysterious, and it would violate currently accepted scientific principles. Strong emergence requires the emergent result to possess new and irreducible causal powers. (See Horgan [9].)

<sup>2</sup> A glider as an emergent phenomenon illustrates that the number of micro phenomena required to produce what are considered emergent macro phenomenon need not be orders of magnitude larger than the number of macro phenomena

---

<sup>3</sup> This is not the same as Prigogine’s [14] notion of dissipative structures, which suggests that a system into which energy is pumped at a moderate level will (almost inevitably) form structures to dissipate that energy. Our conception is that of a non-inevitable process that perpetuates itself by extracting energy from its environment.

<sup>4</sup> One obvious problem with developing an abstract theory of process entities is that it isn’t clear what “stuff” the most primitive processes would process.

reasonable models of the processes that sustain process entities, we will not be able to build models of how these process entities depend on their environments and of how changes to their environments cause their processes either to fail or to be more successful.

A nice example of evolution in the small is the biological arms race. As far as we know, there are no good computer simulations of biological arms races. We believe that the reason for this is that a biological arm race consists of a sequence of creative interventions in the processes of the combatants, i.e., attempts by the two sides either to exploit and or to gum up the other side's processes. Since we can't model architectural processes except on an *ad hoc* case-by-case basis, we can't model how interventions in those processes and evolutionary changes to them may succeed or fail.

Besides the fact that biological processes are very complex, another reason we have difficulty modeling them is that they span the range of physical, chemical, and biological phenomena. Geckos climb walls by exploiting the VanderWaals force, an obscure quantum physics effect. How can we possibly model biological systems if we don't know which fundamental physical forces we can safely leave out of our models?

So here is a fourth problem. We are in deep trouble if we have to build models that include all of physics and chemistry before they can be expected to yield significant results that weren't built into them. What would it take to build an evolutionary model in which gecko climbing emerged?

The following sections discuss these issues in more detail. Although we present some suggestions about how to deal with the issues raised, this paper does not claim to offer complete solutions. Rather, it is an attempt to identify and clarify some of the current problems in our understanding and modeling of complex systems.

Section 2 explores the importance of grounding computation in real world processes; section 3 looks at what fitness in a world of processes really means; and section 4 discusses emergence and the problem of incorporating emergent phenomena back into our computational models.

## **2. NIHIL EX NIHILO: HOW COMPUTER SCIENCE ORIGINATED THE PROBLEM, AND HOW IT HAS PERPETUATED IT**

Nothing operates for free: *nihil ex nihilo*. One can't get a computational process from nothing. To run a computer program (or any computation) in the real world one needs a process powered by a source of energy.

Yet in computer science we typically assume that we can create abstract computational devices or processes from nothing. We say: let  $T = \langle \dots \rangle$  be a Turing Machine, or let  $F = \langle \dots \rangle$  be a finite automaton, or let  $P = \dots$  be a computer program. We then proceed to analyze how that Turing Machine, finite automaton, or computer program would operate.

In other words, we in computer science allow ourselves to presume that we can postulate the existence of fully powered processes such as the Turing Machine—that all we have to do is specify how a particular Turing Machine will operate, and we will have one that operates that way.

### **2.1 Computing as Stigmergy, an Emergent Process**

Turing Machines are fine for analyzing computability, but when we use them as our basic framework for thinking about fundamental real world computational issues we ignore the fact that in reality a computation exists only if there is some energy-driven process that is doing the computing. In the real world, such computing processes are typically provided by the operation of a general purpose computer. Therefore it is worth looking briefly at how a general purpose computer creates a computation.

In simplest terms, a general purpose computer is simply a device that executes individual (machine) instructions, one after another. There are only a finite number of operations<sup>5</sup> that a general purpose computer is capable of performing, and each one is generally quite simple. No one instruction is an algorithm execution<sup>6</sup>. An algorithm execution results when a sequence of instructions is executed. That algorithm execution is not built into the computer. It is an emergent phenomenon in much the same way that a glider is an emergent phenomenon in the Game of Life.

In both cases—a glider and an algorithm execution—one has a simple underlying process—or in the Game of Life a grid of identical processes. The processes that are built into a Game of Life grid are defined by the Game of Life rules. The process that is built into a general purpose computer is its instruction execution cycle. The instruction execution cycle repeats its simple process over and over: fetch an instruction; execute it; fetch another instruction; execute it; etc. Depending on the instructions that the instruction execution cycle encounters, one or another algorithm executions may emerge—just as given the states of the grid cell states that a Game of Life run may encounter, one or another pattern (such as a glider) may emerge.

What's important about this is that in both cases new (emergent) phenomena (either an algorithm execution or a glider) result when existing processes encounter elements in their environments.

The phenomenon of controlling a process by changing the environment in which it operates has been referred to in a computational biology context as *stigmergy*—think ant foraging pheromones. (See, for example, Bonabeau [7].) Although one typically doesn't think of the program that one loads into a

---

<sup>5</sup> If one counts the address field as part of an instruction, and if one wants to ignore the fact that a general purpose computer has finite memory, then one might argue that there are an infinite number of instructions. But no matter how one thinks about memory issues there are still only a finite number of operations. A Turing Machine might be a clearer example of this point in that it has only a finite number of states, i.e., only a finite number of operations that it is capable of performing.

<sup>6</sup> Not just an algorithm, but an algorithm execution. Algorithm executions serve as a useful example because they are both physical processes and emergent phenomena

computer in the same way that one thinks of markers left in a biological environment, they are quite similar. In both cases, these environmental elements affect how the processes that encounter them will behave. In both cases, one is shaping, i.e., programming, an existing process by changing the environment within which that process operates.

## 2.2 Process Programming

We believe that this model—a model in which new processes (such as gliders and algorithm executions) emerge as the environment shapes existing processes—is not only fundamental to computing but that it reflects how the world actually works. We also believe that computer science has done itself (and the disciplines that use its insights) a disservice by not paying more attention to this model.

To illustrate further it is worth noting that a computer's instruction execution cycle may itself be understood as an emergent phenomenon that results when a lower level process is shaped by what it finds in its environment. When building a general purpose computer, computer engineers make use of a pre-existing source of energy in the form of an electrical voltage—or when conceptualized as a process, a flow of electrons. By shaping the environment in which those electrons flow, computer engineers build a framework within which the instruction execution cycle emerges.

Like the Game of Life, whose rules know nothing about gliders, and like the instruction execution cycle, which knows nothing about algorithm executions, the physical laws that control how electrons flow through wires and gates know nothing about computer instructions. But through the clever manipulation of the environment through which electrons flow, computer engineers use the process of electron flows to create an instruction execution cycle process. Like gliders, which are emergent phenomena of the processes that operate in Game of Life cells, and like algorithm executions, which are emergent phenomena of the instruction execution cycle, the instruction execution cycle is itself an emergent phenomenon of an underlying electron flow process.

We believe that the most appropriate model for computing (and for many other physical phenomena) is one in which processes are built up in the manner illustrated above. A model such as this starts with existing fully powered processes (not with nothing) and builds processes on top of them.<sup>7</sup>

## 2.3 Programming as Emergence

Emergence is what results when existing (computational and non-computational) processes are shaped by their environments. The Game of Life is simply a grid of ongoing processes. The rules governing how those processes operate are given; we cannot change what makes Game of Life grid cells turn on and off. But

---

<sup>7</sup> Speculatively, one might be able to build such a model in which the most primitive processes are those in which virtual particles implement the fundamental forces of physics. All other processes would then appear as emergent phenomena that come into existence as previously existing processes are shaped by their environments. Of course, it's not all that simple. For example, it's not clear what the primitive environment is that shapes the primitive processes as they encounter it.

clever (or lucky) hackers that we are, we have discovered that particular configurations of grid cell states result in a glider.

When you think about it, this is quite amazing—so amazing that as we indicated above, Bedau uses the glider as his prototypical example of emergence. Gliders seem amazing because there is nothing in the Game of Life rules that mentions patterns of cell state configurations that travel across unbounded areas of the grid. Yet there is a menagerie of such Game of Life patterns. And as we know (see, for example, Rendell [15]), it is even possible to use such Game of Life patterns to simulate a Turing Machine.

Wolfram [21] uses the same sort of thinking, i.e., setting up conditions that affect how an underlying process will proceed, in his *tour de force* demonstration that a 1-dimensional cellular automaton (CA) running what he calls rule 110 is universal.

Although these computational capabilities seem startling at first glance, when viewed from the perspective of a programmer, neither the glider nor the use of a rule 110 CA for universal computation is all that astonishing. A glider is simple enough. Turn on the right cells within a Game of Life grid, and one gets a glider. And once we know how to use Game of Life processes to produce gliders, we can use those gliders to do computations.<sup>8,9,10</sup>

Since we as computer scientists do this sort of programming for our living we tend to take emergence like this—although not by that name—for granted. We take it so much for granted that in its standard guises, we don't even think of it as emergence. Levels of abstraction and layered hierarchies are emergence in practice. And we are so used to building functional stacks, e.g., for communication, that they seem commonplace. In fact, any executing computer program is an emergent phenomenon. Yet since we build them every day, we find them unremarkable. We may find them charming, entrancing, clever, sometimes beautiful, and perhaps even addicting, but rarely remarkable.

One might generalize this observation and say that any (creative) product that has properties that its components lack is an

---

<sup>8</sup> It is worth noting that there is no glider algorithm: gliders are not the consequence of a traditional computer program, i.e., a sequence of instructions, that explicitly generates them. They result when Game of Life processes are shaped by their environment.

<sup>9</sup> One can even build a library of Game of Life patterns with an API. In doing so, one must take into consideration the details of how the patterns interact, i.e., exactly which pattern cells run into which other pattern cells when two patterns interact. The impossibility of ever fully escaping from lowest level considerations is a continuing theme of this paper.

<sup>10</sup> Since at the pattern level, the Game of Life is Turing complete, it is only partially decidable whether a given pattern will appear when the Game of Life is started from a particular configuration of grid cells. The property of being partially but not totally decidable fits quite well with one of the commonly accepted informal properties of emergence, namely that a phenomenon is emergent if there is no simpler way to determine if it will appear than to run the system in which it may appear and see what happens.

emergent phenomenon.<sup>11</sup> Many human disciplines (the creative arts, engineering, computer science) train their practitioners in emergence. Consequently, we as a society tend to take emergence that results from human activity as relatively commonplace. It is only emergence in nature, emergence that seems to be unplanned, that we find mysterious.

Yet planned or unplanned, the phenomenon is the same: emergence occurs through the shaping of existing processes to create something new—and often new in a way that one would typically not predict by looking at the underlying processes.

We suggest that it is the unpredictable and contingent nature of emergence that Philip Anderson had in mind when in a landmark paper Anderson [4], he contrasted reductionism<sup>12</sup> with what he calls the *constructionist hypothesis* (that the “ability to reduce everything to simple fundamental laws ... implies the ability to start from those laws and reconstruct the universe”), with which he disagrees. He argued that

At each level of complexity entirely new properties appear. ... [O]ne may array the sciences roughly linearly in [the following] hierarchy [in which] the elementary entities of [the science at level n+1] obey the laws of [the science at level n]: elementary particle physics, solid state (or many body) physics, chemistry, molecular biology, cell biology, ..., psychology, social sciences. But this hierarchy does not imply that science [n+1] is ‘just applied [science n].’ At each state entirely new laws, concepts, and generalization are necessary. ... Psychology is not applied biology, nor is biology applied chemistry. ... The whole becomes not only more than but very different from the sum of its parts.

### 3. FITNESS

If our world is a world primarily of processes (rather than of things), a central question is what keeps the processes running. Clearly the answer is that processes operate only when they have access to sufficient energy and materials to maintain themselves. Our basic model then is of a world of process entities (not necessarily biological) that must “consume” (in some loose general sense) energy and materials that they find in their environment in order to sustain and perpetuate themselves.

A relatively simple (and non-biological) real-world example of a self-sustaining process that uses (and depends on) energy and materials that it extracts from the environment is a hurricane. Air and water vapor flow through a hurricane, which persists only as long as it can extract sufficient energy and materials from its environment. Here is a description NASA [12] of how hurricanes work.

As [warm moisture-laden surface air] rises, it expands and cools triggering ... condensation, [which results in] the release of ... latent

<sup>11</sup> Having properties that its components lack is our definition of emergence in Abbott [1].

<sup>12</sup> Anderson defined reductionism as the assumption that the “workings of all the animate and inanimate matter of which we have any detailed knowledge are all ... controlled by the same set of fundamental laws [of physics].” Anderson does not reject reductionism. “[W]e must all start with reductionism, which I fully accept.”

heat, and an ... increase in buoyancy, thus allowing more air to rise. A chain reaction (or feedback mechanism) is now in progress, as the rising temperatures in the center of the storm cause surface pressures to lower even more. Lower surface pressures encourage a more rapid inflow of air at the surface, more thunderstorms, more heat, lower surface pressure, stronger winds, and so on.

Meanwhile air pressures near the top of the storm, in response to the latent heat warming, ... rise. In response to higher pressures aloft, air begins to flow outward (diverge) around the top of the center of the cyclone. Analogous to a chimney, this upper-level area of high pressure vents the tropical system, preventing the air converging at the surface from piling up around the center. [See Figure 1.]

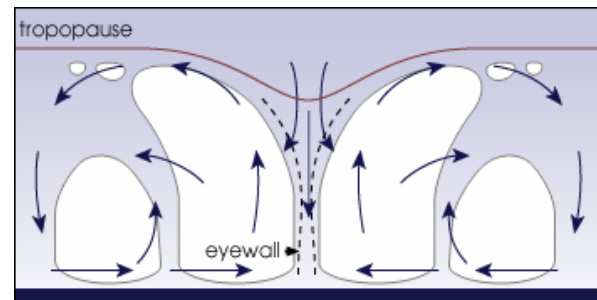


Figure 1. Anatomy of a hurricane

As this description indicates, the environmental resources upon which hurricanes depend are (a) the energy that transfers (b) moisture from the ocean to the relatively warm (c) surface air before it is pumped upward and (d) the continual dispersal of heat in the upper atmosphere so that the heat generated by condensation does not overly warm the condensation area. With these environmental conditions in place, hurricanes can perpetuate themselves indefinitely.<sup>13</sup>

Although one should not make too much of this, it is worth noting that like a biological system a hurricane generates heat internally—although interestingly enough, heat generation occurs in the upper atmosphere which is where condensation takes place. A hurricane is a form of heat engine. But unlike heat engines that we build as human artifacts, a hurricane is self-organizing and self-perpetuating; no external maintenance is required. Although hurricanes generate heat by condensation rather than by oxidation, they take in moist air as an energy resource and expel dryer air as a waste product. The primary work that is done by a hurricane-as-heat-engine is to move the air that powers it.

Hurricanes persist as long as the environments within which they find themselves continue to provide the needed resources. One might talk about the fitness of a hurricane to its environment as the degree to which the environment supplies those resources. When a hurricane moves out of an environment that suits it, e.g., to one with a cooler ocean surface or over land, it is no longer able to sustain its internal process, and it dies. This is quite similar to our sense of the fitness of an organism to its environment.

<sup>13</sup> Would that we had similarly comprehensive descriptions of how (even simple) biological organisms perpetuate themselves.

Of course hurricanes don't have a mutable genetic code—obviously they don't have a genetic code at all—and hurricanes don't reproduce. So hurricanes don't evolve. But in the sense in which one can speak about the fitness of a hurricane to its environment, hurricanes are quite like other entities that find themselves in an environment for which one can say they are more or less fit.

When viewed on a longer time scale, hurricanes are historical and contingent elements of the world. To the extent that the environment supports them, there will be hurricanes. If the environment did not support them, they would not exist. And just as there are biological species that depend on forest fires, presumably there are species that depend on hurricanes for their existence—although as non-biologists, we don't know what they are. Thus hurricanes form part of an ecosystem; they create a niche within which other elements of the ecosystem live. As contingent historical entities, hurricanes may be said to play a role in evolutionary history similar to that played by a biological species.

Again, without making too much of it, we are stressing the similarity between hurricanes and biological entities as a way of pointing out that the concept of fitness that we are urging is one that applies to any self-perpetuating process, biological or not.

The fitness (or lack thereof) of an entity to its environment is a fundamental feature of evolutionary biology. Yet fitness cannot be modeled in real terms unless one is able to model how an entity depends on and makes use of the resources in its environment. We argue that entities are fundamentally processes that depend on their environment for the energy that powers them and for the materials that they use to structure and organize themselves. If we don't model entities as processes and if we don't model how those processes depend on their environments, we will not be successful in modeling important evolutionary aspects of the world.

Populations evolve when they find new ways of extracting and exploiting the energy and materials they find in their environment, i.e., when they generate more clever processes. From the perspective in which it is primarily processes rather than structures that evolve, evolution isn't so much a blind watchmaker as a blind programmer, one that generates processes that are suited to their environments.

## 4. THE REALITY OF EMERGENT EPIPHENOMENA

There is a fundamental difference between how newly developed computer programs function in the world and how emergent phenomena in our computer models fail to function in those model environments.

As we said above, every new computer program is an emergent phenomenon. Yet once deployed, new computer programs take their place in the world and function just like everything else. Once one writes and deploys a piece of software, there it is, as much a part of one's world as anything else.

The same is not true of emergent phenomena in our computer models. Consider a glider again. Gliders do not participate in a Game of Life simulation at the same level as the rules of the Game of Life. We as programmers can make use of gliders, but gliders don't enter the realm of a Game of Life simulation in anything

like the way that newly written and deployed computer programs enter the realm of processes that users experience.

The same, as we said earlier, is true of "boids" and virtually every other phenomenon that we take as emergent within a computer-implemented model. These emergent phenomenon do not re-enter the simulation from which they emerged as operational elements. We as outside observers may see phenomena emerge, but elements within the simulation don't.

An important feature here involves the ways in which elements of a simulation interact with each other. In most simulations, elements interact with each other according to rules that are built into the simulation. But when a new type of element emerges, there are no rules for how other elements are to interact with it. Consequently, the built-in elements of a simulation can't and don't interact with emergent phenomena in terms of rules. Nor do emergent phenomena interact with each other in terms of rules. As a result emergent phenomena tend to retain their epiphenomenal (see discussion below) and shadowy quality: they are there, but not really.

This will remain the case until we develop systems that can recognize emergent phenomena as they emerge, incorporate them into the system as new element types, and generate (or identify) rules for how they interact with each other and with previously existing element types. Until then, emergent phenomena in computer simulations remain ideational; they exist primarily (only?) in the minds of the observers. The computer model does not include rules for how those phenomena interact either with each other or with elements that were built into the model originally. Yet since the model does contain rules for how its primitive elements interact, emergent phenomena always exist as second class citizens, observable to those of us outside the model who can see the model run, but invisible to anything within the model, even to themselves and to each other.

### 4.1 Noumena, Phenomena, Epiphenomena, and Constructionism

In this subsection, we step back a bit and discuss the distinction between the terms *phenomena* and *epiphenomena*. We also discuss some of the implications of that distinction.

A good place to start is with Kant [10], who coined the term *noumena* to refer to *the thing in itself*, i.e., reality as it is and independent of our perception or understanding of it.

One step away from the noumena, the term *phenomena* may be understood to refer to aspects of reality that we are able to capture within our perceptual or conceptual framework, i.e., anything that we can apprehend using human perception, understanding, or the tools of science.

One of the issues that we are raising in this paper is the difficulty (perhaps impossibility) of establishing a phenomenological (or ontological) framework that will be satisfactory for what one wants to model, i.e., the gecko problem. It may be that no phenomenological framework is satisfactory for modeling all biological phenomena. It may also be that we will never construct a phenomenological framework that captures all of the noumena.

The term *epiphenomena* is best understood as referring to aspects of phenomena. Epiphenomena are often referred to as secondary.

For example, The American Heritage Dictionary [3] definition of *epiphenomenon* reads as follows.

A secondary phenomenon that results from and accompanies another: “Exploitation of one social class or ethnic group by another [is] an epiphenomenon of real differences in power between social groups”

The term *by-product* is also frequently used when defining *epiphenomenon*. But *by-product* is used not to refer to a consequence of some other phenomenon but to refer to some other way of apprehending or perceiving the phenomenon.

Epiphenomena are typically understood to add no causal power to existing phenomena. Gliders may be said to be epiphenomena of the Game of Life rules: they appear as a by-product or secondary consequence of the operation of those rules. (The primary consequence is simply the switching on and off of the grid cells.)

The behavior and properties of gliders are totally explicable in terms of the Game of Life rules, i.e., gliders may be explained reductively in terms of the Game of Life rules. Furthermore, gliders have no causal power within the Game of Life. Their existence has absolutely no effect on how the rules operate or on what results the rules produce, i.e., which cells will be switched on or off. A Game of Life run will turn out exactly the same whether one notices the gliders or not. In some sense, then, epiphenomena such as gliders are entirely ideational.<sup>14</sup>

This brings us to another issue that we are raising in this paper: the apparent concreteness of epiphenomena in the real world. Although epiphenomena can be explicated in reductionist terms, in the world as we know it they tend to fold themselves back into the world and to add something real and new. As mentioned above, although gliders are epiphenomena of the Game of Life rules, they can also be used to do real computations, i.e., to simulate a Turing Machine. One can even create what might be called a programming library of Game of Life patterns which includes descriptions of how the patterns interact with each other. When treated in this way, patterns, which originate as epiphenomena of Game of Life rules, take on a concreteness that allows us to treat them as real objects—objects that interact with each other in terms of well-defined interfaces.

A strict reductionist might argue that Game of Life patterns are illusory nonetheless. After all, the Game of Life consists of nothing but a grid of cells that blink on and off. There is no such thing as a glider. At best, gliders are conceptual conveniences and mental shortcuts.

But that same argument could be applied to virtually everything in our daily lives. To say that everything is illusory is to say that nothing is illusory since the term would have no meaning. Moreover, to relegate everything to the land of illusion is to give up on the possibility of understanding relationships among these illusions—unless again, the term illusory loses its meaning. Unless we want to forgo the possibility of understanding the world in terms of objects that we recognize, i.e., as something more than a collection of quarks or strings or branes,

epiphenomena must be understood as having a concreteness that allows us to talk about them and their inter-relationships..

This issue was explored nearly forty years ago by Sperry [18] when he asked how, if one limits oneself strictly to fundamental physical phenomena, would one explain the trajectory of a molecule on the rim of a wheel (a non- fundamental entity) as it rolls downhill. An explanation in terms of fundamental forces—which would have to include a description of how the wheel holds itself together—would be extraordinarily complex at best.

Even more interesting might be the same question but with the wheel rolling downhill replaced by the toenail of a fox as it chases a rabbit. What would an explanation of that trajectory look like when expressed in terms of fundamental physical forces?

Or consider asking for an explanation, also in terms of fundamental physical forces, of the fact that a small black dot appears at the end of this sentence (whatever *sentence* means)—and that it does so in all copies (whatever *copies* means) of this paper (whatever *this paper* means)—either in print or on a computer screen and however it is formatted, e.g., one column or two, using Times Roman or Arial font, etc.. These identical dots are certainly not due to quantum entanglement.

Even if one had explanations of these phenomena in terms of fundamental physical forces, of what good would they be? They certainly wouldn’t illuminate in any useful way the phenomena being explained.

It was presumably this sort of issue that prompted Anderson to reject what we might call *constructionism*, i.e., the constructionist hypothesis.

## 4.2 Entities

The approach that we proposed in Abbott [1] to dealing with this problem is to focus on entities. We tend to see the world in terms of entities. The question is: when is it reasonable to say that some aspect of the world is an entity—rather than, for example, simply a collection of (lower level) components. As we said earlier, our solution was to establish as a criterion for an entity that there be a property (a) that applies to the putative entity and not to its components and (b) that depends on the forces that hold the putative entity together. This seems to work well both for low-level physical entities at an energy equilibrium and for far-from-equilibrium entities.

Another way of approaching the question of when to recognize the existence of an entity is to say that an entity exists when one can draw a boundary within which a reduced entropy level is maintained. Again, this works well for fundamental physical entities such as atoms and molecules. In these cases, entropy is reduced and remains reduced as a result of an energy equilibrium. It also works well for far-from-equilibrium systems in which the process through which the system perpetuates itself, although one that uses external sources of energy, keeps the internal entropy lower than that of the environment. This approach even works well for man-made artifacts—although the process that maintains the reduced level of entropy, i.e., “maintenance,” is applied from the outside, i.e., it is extrinsic to the entity rather than intrinsic as in the other cases.

At this point it is not yet clear how to apply the notion of entropy to processes. The obvious notion is that the processes should be required to reduce entropy. I don’t know how to extend that to a

---

<sup>14</sup> A philosophical term used for this sort of situation is supervenience. Gliders and other Game of Life patterns supervene over the Game of Life rules. For a very nice discussion of supervenience and emergence see Seager [17].

Game of Life glider. Certainly a glider is a low-entropy pattern in a Game of Life run. But I don't know how to say that in better terms.<sup>15</sup> Another approach would be to use Chaitin<sup>16</sup> complexity to show that a process is a shorter way of expressing information about a pattern than the pattern itself.

### 4.3 What Makes a Good Explanation?

In this section we step back from the question of what is real and focus on what terms we want to use to explain the world to ourselves. If we are to understand the world, explications of its functioning must be given in terms that we can use to build intuition, insight, and understanding. And to provide such an explanation means to provide an explanation in terms of the relevant ontological entities.<sup>17</sup>

Quantum theory notwithstanding, and whether one agrees with our particular criteria for entities, it seems essential to focus on epiphenomenal entities as a way of avoiding the constructionist hypothesis. Without higher level entities, the only elements among which one can define relationships are the most primitive elements of physics. When one works with higher level entities, one at least gives oneself the possibility of describing interactions at those higher levels.<sup>18</sup>

A strict reductionist might still argue that no matter which entities one certifies, it is still all quarks (or strings or branes) underneath. A Game of Life configured to simulate a Turing Machine is still doing no more than (just) running the Game of Life rules.

That is true. Denial of constructionism is not denial of reductionism. Denial of constructionism is simply the stance that one cannot fully understand (in terms that we take as useful) higher level interactions by looking only at lowest level phenomena. Once higher level entities appear in the world, they add their own logic to how the world operates, a logic that can't be explicated strictly in lowest level terms. And if one wants to understand (which is the premise of this section) the higher level functioning of a system, one must think about that system in terms of higher level entities, whether or not those entities are epiphenomenal over or supervene over some lower level.

Imagine running a complex computer simulation that displays its progress by means of an ongoing screen animation. Then imagine that someone asks for an explanation of that animation. Suppose

<sup>15</sup> The term *edge of chaos* has been used in the past for similar phenomena. A Google search of "*edge of chaos*" *entropy* returns more than 6,000 references.

<sup>16</sup> See Chaitin's home page for numerous references: <http://www.umcs.maine.edu/~chaitin/>.

<sup>17</sup> Quantum theory suggests this is not quite true. Quantum theory offers a way to compute predictions that are amazingly accurate even though no one seems able to develop an intuitive conceptualization of what quantum theory means. As Richard Feynman famously said, "I think I can safely say that nobody understands quantum mechanics." [8]

<sup>18</sup> One might argue that this is setting up a straw man, that reductionist explanations typically attempt to explain one phenomenological level in terms of the next lower level, e.g., explain biology in terms of chemistry. But if each level can be so explained, all one is eventually left with is the stuff at the lowest level.

that one claimed to provide such an explanation by offering a detailed description of electron flows within the computer and phosphor excitation states on the screen. Although such a description would be a complete explanation of how the system functions, it obviously would not be satisfactory. No insight or understanding is provided when one explains a computer animation in terms of electrons flows and phosphor excitation states.

What makes a good explanation? Suppose that we are Game of Life "naturalists" who are examining various specimens of Game of Life runs found "in the wild." We do not know what the Game of Life rules are. Some of the Game of Life executions include gliders, which we find fascinating because they seem to traverse the Game of Life grid. How can all this be explained?

As good scientists, one approach we take will certainly be to take apart one of our Game of Life specimens and see if we can figure out how it operates. After considerable analysis, we deduce the Game of Life rules. This certainly is a significant accomplishment, and we are justly proud to publish our findings. We are especially happy that we can show that whenever any of a particular set of grid cell configurations occurs, a glider will follow.

Is that the end? No. One of our specimens is a Game of Life execution that is simulating a Turing Machine. As a result of our analysis we can explain every step in that Game of Life execution. What we can't explain is why this Game of Life run is doing what we see, which seems somehow unusually orderly. The Game of Life rules that we discovered can't tell us that some curious teenage hacker had set up that run because he wanted to demonstrate that the Game of Life could simulate a Turing Machine. Our analysis can't tell us that the gliders were established in the position where we found them because our teenager knew how gliders and other patterns interact. In other words, what is missing from our explanation is an explanation of how that particular Game of Life execution works at the level at which it was designed. And without that level of explanation, we are missing an essential element in our understanding of that and presumably other Game of Life runs.

Does that sound too teleological, that what we claim is missing from our explanation is a goal-based description of our entity? In this case, we are presuming a designer/programmer, and we are presuming that the designer/programmer had an intent, i.e., to simulate a Turing Machine. So not understanding that intent or how it might be realized in a Game of Life run is a major hole in our understanding of our Game of Life specimens.

But evolution is also a designer/programmer, albeit a blind one, and although *intent* is not the right word, designs survive (both in "the wild" and in society) if they work. An explanation of how an entity functions without an explanation at a relevant level of how its design works is simply incomplete.

Those of us who have taught computer programming have encountered a similar problem when we talk about software documentation. What is it that one documents when one documents software? Consider the following instance of a familiar software idiom.

```
temp := x;  
x := y;  
y := temp;
```

A comment such as the following is clearly not satisfactory documentation.

```
// Store x in temp.  
// Then assign y to x and temp to y.
```

Not only does this comment not tell us anything that is not already visible in the code, it doesn't tell us anything about why the code is doing what it does. Obviously what we want is something like the following.

```
// Exchange x and y.
```

Why do we want to know this? The exchange of *x* and *y* is the epiphenomenal effect of the three lines of code, a phenomenon that requires all three lines of code to occur. Although it is epiphenomenal—*x* and *y* will be exchanged whether a comment says so or not.—we believe that this sort of information is important when documenting software.

What would be even more important would be a comment such as the following were code such as this part of a simulation of an agent-based bartering system.

```
// In these steps x and y consummate their  
// bartering agreement by exchanging assets.
```

So clearly we think of design at all levels as important in understanding phenomena. Explanations that don't explain how designs function are not adequate.

Does this sound too trivial? Are we saying that a fundamental question of how to understand the world is comparable to the question of how to document software? In fact we are. In many ways software is thinking made concrete. Many of the question that software developers have had to face are issues that we never faced squarely until now. These are issues of ontology and semantics that we have heretofore been able to avoid because we did not have the tools to externalize our thoughts in anywhere near the detail that we do now.

Software can be *about* nearly anything. So determining how to document software, i.e., how to describe what the software is about, raises very broad philosophical questions. UML [13] and OWL [19] are two current (and only partially successful) attempts to formalize ways of representing knowledge about what software is about.

There is an even more abstract point to be made. Let's consider once again a Game of Life run that uses patterns such as gliders to simulate a Turing Machine. Consider the following question: can one prove that such a system simulates a Turing Machine without using concepts such as gliders?

Presumably each step in the argument that the construction does simulate a Turing Machine could be restated directly in terms of Game of Life rules and grid cells. After all, nothing is really happening other than the application of Game of Life rules. But what would such a restatement look like? What would the restatement be able to claim is being done?

To prove Turing universality, one must define a mapping from a Game of Life process to a Turing Machine and then demonstrate that when the action of the Game of Life grid cells are understood in terms of the mapping, the result is equivalent to a Turing Machine computation. It seems to me that this is really no different from speaking in terms of Game of Life patterns. It just sounds a bit more formal.

The point is that if one wants to prove a connection between a Game of Life process and a Turing Machine, one is forced to map one onto the other. Moreover, the *concept* of a Turing Machine can't be avoided since it is that concept that is at the heart of the proof. Yet once one allows oneself to speak in terms of anything other than Game of Life grid cells, one is talking about phenomena that are epiphenomenal to the Game of Life.

So it would seem that unless one allows oneself to think in terms of epiphenomena, one is very limited in terms of what one can say. Without operating on the epiphenomenal level one cannot prove the Turing universality of the Game of Life.

More generally, whenever we want to link abstractions to reality, we must talk in terms of epiphenomena. Until it is linked to reality, an abstraction is nothing more than something that goes on in people's mind.<sup>19</sup> Once we connect an abstract thought to a real physical process, the thought becomes an epiphenomenon of the process—assuming that the connection is successful. The thought is epiphenomenal of the reality because the process does its thing whether the thought is linked to it or not.

A consequence of this is that if we as human beings are to understand the world, i.e., to represent the world in terms that work in our minds, we are forced to work in terms of epiphenomena. Once we realize this, it would seem that on grounds of convenience alone and ignoring the issue of the reality of epiphenomena, we should embrace descriptions of the world, as long as they are successful, in terms that work best for our minds, no matter at what level those descriptions are expressed.

#### 4.4 Emergence as a Contingent Historical Process

A fundamental difference between emergent phenomena in the real world and emergent phenomena in computer systems is that computer systems tend to be organized (in fact we pride ourselves on organizing them) as encapsulated and stratified levels of abstraction or layered hierarchies. The best way to design (and once designed to understand) a Game of Life run configured to simulate a Turing Machine is in terms of layers: the lowest layer is the Game of Life rules running on a grid; the next layer is the library of patterns that a Game of Life run is capable of producing; the highest layer is the Turing Machine that is built using these patterns.<sup>20</sup>

The real world is not organized into disjoint and stratified layers. There is only one real world, and it is not a layer cake of

---

<sup>19</sup> In saying this we are deliberately sidestepping the problem of what we mean by consciousness and conscious thought. We are assuming that subjective experience, i.e., qualia, thoughts, etc., comprise an area that we don't yet know how to explicate. We are also assuming that a concept in the mind, whatever that means, is not the same as the referent, if any, of that concept. Unless one is a thoroughgoing Platonist, some concepts don't have real-world referents. Other than more concepts, what is the referent of the concept Turing universality?

<sup>20</sup> Many software systems are not as hierarchical as we tell ourselves. We use class libraries, but the classes in these libraries are not strictly stratified. In our Turing Machine example, there may be a Turing Machine pattern that takes its place in the library along with the glider pattern.

independent worlds joined together by dabs of interface filling and icing. Emergence is the result of the combining, selecting, and shaping existing processes by the environment to build new processes. These new processes then become part of the world and are available, along with everything, else to be combined and shaped into even newer processes. Because all this activity occurs (a) in an integrated world of interacting phenomena and (b) as a result of what are often arbitrarily circumstantial environmental factors, and because these processes occur in time, i.e., as some sequence of events, emergence is necessarily a contingent historical process.

Once one gets past the formation of entities by finding equilibrium states, the rest is contingent. How processes will be shaped, how they will combine, which new ones will emerge, and which will survive, all depend on both accidents of the environment and the creativity of (intentional or blind) programmers. Since emergence is a consequence of how processes interact with their environment, since some processes are fundamentally probabilistic (as well as chaotic), and since the environment itself includes previously formed processes, there is a contingent and historical quality to emergence that cannot be avoided by appeals to reductionism. Emergence is a forward-looking process in which entities, having emerged, take their place in the world that begot them, and in so doing change that world. In other words, the very ontological nature of the world is contingent and historical rather than structured and hierarchical.

We rely on this process of creative emergence as we build increasingly sophisticated software systems—and, in fact, as we build an increasingly sophisticated society.<sup>21</sup> Would that we could model it in a simulated environment.<sup>22</sup>

#### 4.5 A River as an Emergent Phenomenon

We would like to end with an interesting (although perhaps somewhat more difficult) real-world example of an emergent phenomenon: a river. If one looks at the fundamental laws of chemistry and physics, there is no such thing as a river—just as there is no such thing as a hurricane. Rivers come to exist as an emergent epiphenomenon of the laws governing gravity, evaporation, condensation, the behavior of gases under various temperature and pressure conditions, etc.<sup>23</sup> A river is just a

<sup>21</sup> A nice example of a social process building on other processes is the emergence of a market for virtual resources in online games. (See, for example, BBC [5].) Apparently companies have established businesses in which low-wage Mexican and Chinese teenagers are paid to play games and earn resources that are then sold on eBay.

<sup>22</sup> The closest we have come is genetic programming. But under that paradigm evolution takes place outside the operational environment; programs do not evolve within the environment within which they function. Another concern with genetic programming is its reliance on fitness functions rather than on the actual capacity of elements to make use of resources in the environment. (See Abbott [2] for a further discussion of this issue.)

<sup>23</sup> More narrowly focused, one might say that a river is simply an emergent phenomenon of the force of gravity as constrained by geological features operating on a continually renewed supply of water.

continually re-supplied downhill flow of water molecules. If one applies the laws of physics to the basic geological and weather features of the planet, rivers will presumably be among the phenomena that emerge. The key point, however, is that a river is not a thing, it is water molecules in motion. As a persistent feature of the world, a river is epiphenomenal

But from the perspective of its biological inhabitants, a river is a persistent feature of the environment. As far as these creatures are concerned, the river persists—even though at any given moment different water molecules are flowing through it.

It is just as true to say that a river persists even though the water flowing through it changes as it is to say (a) that a government persists even though the individuals who hold particular offices in that government change and (b) that a person persists even though the materials of which we are made cycle through us. In all three cases (and many others) it is a process or a collection of processes that persist, not the particular materials with which the processes are operating at any particular time. All three examples are epiphenomena.

Once a river emerges as a persistent if epiphenomenal process, it is able to support further emergence, e.g., of an ecology within its flow and along its banks. Elements of that ecology, such as a beaver dam, may then turn around and change the river itself. It is in this sense that the status of a river changes from being an epiphenomenal consequence of more primitive processes to being a real component of the physical world.

We have already rejected the argument that as an epiphenomenon a river is just an illusion. But if we reject that position, we are faced with the question of how to allow epiphenomena such as rivers to insert themselves back into our computer models. I know of no modeling or simulation system that allows that. Yet it is clearly a natural and essential aspect of the world in which we live.

From rivers to software systems to markets to bureaucracies to nation-states, most of what we think of as essential features of the world around us consists of epiphenomenal processes. Yet we experience them as real. How is it that we manage this so easily in the real world and have so much difficulty with it in simulated worlds?

## 5. SUMMARY

Ten years ago, Steven Weinberg used [20] the weather as an example to make the anti-emergence anti-entity case, the case for what might be called extreme reductionism.

“[T]he reductionist view emphasizes that the weather behaves the way it does because of the general principles of aerodynamics, radiation flow, and so on (as well as historical accidents like the size and orbit of the earth), but in order to predict the weather tomorrow *it may be more useful to think* about cold fronts and thunderstorms. Reductionism may or may not be a good guide for a program of weather forecasting, but it provides the necessary insight that *there are not autonomous laws of weather that are logically independent of the principles of physics*. Whether or not it helps the meteorologist to keep it in mind, cold fronts are the way they are because of the properties of air and water vapor and so on which in turn are the way they are because of the principles of chemistry and physics. *We don't know the final laws of nature, but we know that they are not expressed*

in terms of cold fronts or thunderstorms.” [emphasis added in all cases]

Certainly there are no autonomous laws of weather that are logically independent of the principles of physics. If there were we would have a case of strong emergence. The question is whether concepts such as cold fronts and thunderstorms are just convenient ways of thinking, i.e., arbitrary though practical constructs, or whether we are justified in saying that they exist as real entities in the world. What is their ontological status. Do they really exist, or are they just conceptual or useful conveniences? Our view is that it is not anti-reductionist to suppose that higher level entities really exist; it is anti-constructionist to deny that they do.

Our case has been that entities such as these are real in the same way that gliders and other patterns in the Game of Life are real. These elements represent localized portions of the world with reduced entropy. And just as one can develop a library of Game of Life patterns with a well-defined API, one can develop a science of thunderstorms and cold fronts.

Furthermore, without such a science, our understanding of the weather would be much poorer. To throw out gliders and cold fronts and simply replace them with lower level explanations is to deny ourselves a complete understanding of aspects of reality in much the same way as we would be denying ourselves an understanding of the functioning of software if we thought about it only at the level of electrons moving about.

However, we do not yet know how to formalize the approach that we believe is necessary. We expect that it will require a fair amount of work along the following lines.

1. What's needed most fundamentally is a theory of process programming that clarifies how computing is done in terms of existing physical (i.e., energy/entropy-driven) processes and how new processes are shaped by the interaction of existing processes with their environment. This is not a new thought (see, for example, Landauer [11]), but as far as we know, it has not been taken very far.
2. Also needed is a more complete theory of entities and a fuller understanding of how to develop ontological frameworks that don't reductively collapse on themselves but that support the addition, as first-class elements, of new and unpredictable entities as they emerge.
3. Finally, what is needed are computational techniques that will allow us build computer systems within which we can experiment with models that embody solutions to these problems.

## 6. ACKNOWLEDGEMENT

I am grateful for numerous enjoyable and insightful discussions with Debora Shuger during which many of the ideas in this paper were developed and honed.

## 7. REFERENCES

- [1] Abbott, R., "Emergence, Entities, Entropy, and Binding Forces," *The Agent 2004 Conference on: Social Dynamics: Interaction, Reflexivity and Emergence*, Argonne National Labs and University of Chicago, October 2004. URL as of 4/2005:

[http://abbott.calstatela.edu/PapersAndTalks/abbott\\_agent\\_2004.pdf](http://abbott.calstatela.edu/PapersAndTalks/abbott_agent_2004.pdf).

- [2] Abbott, R., J. Guo, and B. Parviz, "Guided Genetic Programming," *Proc of Sixth International Conference on Computational Intelligence and Natural Computing, September 2003*; an earlier version is in *The 2003 International Conference on Machine Learning, Models, Technologies and Applications (MLMTA)* June 2003. URL as of 4/2005: <http://abbott.calstatela.edu/PapersAndTalks/Guided Genetic Programming.pdf>.
- [3] American Heritage Dictionary of the English Language, 4<sup>th</sup> edition, Houghton-Mifflin. 2000. URL as of 11/2004: <http://www.yourdictionary.com/ahd/e/e0180800.html>.
- [4] Anderson, P.W., "More is Different," *Science*, 177 393-396, 1972.
- [5] BBC News, "Gamer buys \$26,500 virtual land," BBC News, Dec. 17, 2004. URL as of 2/2005: <http://news.bbc.co.uk/1/hi/technology/4104731.stm>.
- [6] Bedau, M.A., "Downward causation and the autonomy of weak emergence". *Principia* 6 (2002): 5-50. URL as of 11/2004: <http://www.reed.edu/~mab/papers/principia.pdf>.
- [7] Bonabeau, E., M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, US, 1999.
- [8] Feynman, R. *The Character of Physical Laws (The Messenger Lectures 1964)*, MIT Press, 1967. URL of excerpt as of 4/2005: <http://bouman.chem.georgetown.edu/general/feynman.html>.
- [9] Horgan, T., "Replies to Papers," *Grazer Philosophische Studien* 63 (2002), 303-41. Issue on the philosophy of Terence Horgan. URL as of 11/2004: <http://dingo.sbs.arizona.edu/~thorgan/papers/Replies.to.Papers.htm>.
- [10] Kant, E., *The Critique of Pure Reason*. 1781 and 1787. URL (translation) as of 3/2005: <http://www.arts.cuhk.edu.hk/Philosophy/Kant/cpr/>.
- [11] Landauer, R., *Fundamental Physical Limitations of the Computational Process*, National Bureau of Standards, Publication. SP-614, 1981.
- [12] NASA (National Aeronautics and Space Administration), "Hurricanes: The Greatest Storms on Earth," *Earth Observatory*, 2004. URL as of 3/2005 <http://earthobservatory.nasa.gov/Library/Hurricanes/>.
- [13] Object Management Group, Unified Modeling Language (UML), URL as of 4.2005: <http://www.uml.org/ - UML2.0>
- [14] Prigogine, Ilya and Dilip Kondepudi, *Modern Thermodynamics: from Heat Engines to Dissipative Structures*, John Wiley & Sons, N.Y., 1997.
- [15] Rendell, Paul, "Turing Universality in the Game of Life," in Adamatzky, Andrew (ed.), *Collision-Based Computing*, Springer, 2002. URL as of 4/2005: <http://www.cs.ualberta.ca/~bulitko/F02/papers/rendell.d3.pdf> and [http://www.cs.ualberta.ca/~bulitko/F02/papers/tm\\_words.pdf](http://www.cs.ualberta.ca/~bulitko/F02/papers/tm_words.pdf)

- [16] Reynolds, C. W. "Flocks, Herds, and Schools: A Distributed Behavioral Model," in *Computer Graphics*, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34, 1987. URL as of 3/2005: <http://www.cs.toronto.edu/~dt/siggraph97-course/cwr87/>.
- [17] Seager, W. "Supervenience and Emergence", draft. URL as of 4/2005: <http://www.utsc.utoronto.ca/~seager/emsup.pdf>.
- [18] Sperry, R.W. 1969. "A modified concept of consciousness," *Psychol. Rev.* 76: 532-536.
- [19] W3C, *OWL Web Ontology Language*, URL as of 4/2005: <http://www.w3.org/TR/owl-ref/>.
- [20] Weinberg, S., "Reductionism Redux," in Cornwell, J. (ed), *Nature's Imagination: The Frontiers of Scientific Vision*, Oxford University Press, 1995.
- [21] Wolfram, S., *A New Kind of Science*, Wolfram Media, 2002. URL as of 2/2005: <http://www.wolframscience.com/nksonline/toc.html>.