

1/21/05

Evolution as Computation

Evolutionary Theory (accepted for publication)

By: John E. Mayfield

Program in Bioinformatics and Computational Biology

Department of Genetics, Development, and Cell Biology

2106 Molecular Biology Building

Iowa State University

Ames IA 50010

Phone: 515-294-6847

Email: jemayf@iastate.edu

Key words: Evolution, Computation, Complexity, Depth

Running head: Evolution as Computation

Abstract

The existence of genetic and evolutionary computer algorithms and the obvious importance of information to life processes when considered together suggest that the formalisms of computation can provide significant insight into the evolutionary process. A useful computational theory of evolution must include life and also non-living processes that change over time in a manner similar to that observed for life. It is argued in this paper that when the evolution of life is partitioned into a DNA part and a soma or phenotype part, that the DNA part may be properly treated as a computation. A specific class of computations that exhibit biology-like evolution is identified to be the iterated probabilistic computations with selection when selection is based on interactions among evolving entities. When selection is not based on interaction with other evolving entities, the term optimization seems more appropriate than does evolution. One implication of the view presented in this paper is that fundamental aspects of evolution derive from and depend upon properties of information. Another consequence is that the mathematics of computation applies directly to the evolution of DNA. The paper introduces an information like measure of complexity, transcript depth, which incorporates the minimal computational effort invested in the organization of discrete objects. Because the formation of organisms relies on information stored in their DNA, organism complexity is constrained in a fundamental way by DNA transcript depth.

Introduction

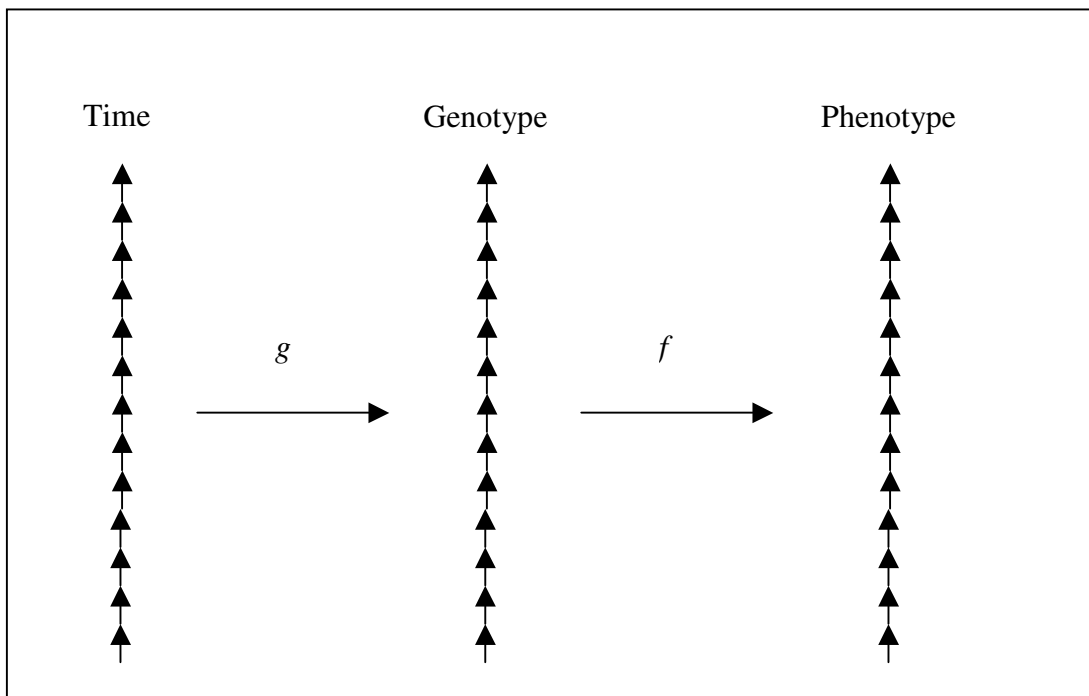
The Darwinian paradigm has long been recognized as algorithmic. Discovery in the 1960's and 1970's that principles borrowed from biological evolution could be encoded in computer algorithms that "evolved" in ways reminiscent of what happens in biology was a great revelation to some, and spawned a new field of investigation (Holland, 1975; Goldberg, 1989; Fogel, 1998). These discoveries contribute to a wide range of disciplines through the application of genetic and/or evolutionary algorithms. The general idea of taking concepts familiar to evolutionary biologists and applying them in non-biological contexts is invoked regularly in diverse fields. This is a tribute to the explanatory power of Darwin's basic idea. Frequently the word evolution is used as a synonym for change, but often the author means something more specific. Not uncommonly the intended meaning is "biology-like" evolution, recognizing that the Darwinian paradigm provides useful insight into a variety of non-living systems. Given the wide range of phenomena that are described as evolutionary in this specific sense, it is surprising that little effort has been devoted to defining biology-like evolution. This paper proposes such a definition and explores some of the implications.

Stadler et al. (2001) formally describe biological evolution as a trajectory composed of two maps (figure 1). The first map, $g_n(g_{n-1}, t)$, assigns a [discrete] genotype to each point in [discrete] time, and the second, $f_n(f_{n-1}, g_n)$, assigns a phenotype to each genotype. Map g describes a one-to-many relationship between time and genotype, while map f describes a many-to-many relationship between genotype and phenotype. Both maps may be defined to accept environmental and/or stochastic input. This scheme formalizes a widely held understanding of how life changes over time (measured in generations). The scheme effectively partitions the analysis of life into the

evolution of DNA (g map) and the growth and development of organisms (f map). Stadler et al. focus on the implications of this formalization for understanding phenotype changes over evolutionary time. As such, their focus is on the f map. The present paper is concerned primarily with the g map.

Figure 1

Evolutionary trajectory decomposed into two parts.



Generalizing the two part evolutionary trajectory depicted in figure 1 to include non-living systems requires generalized definitions of genotype and phenotype and a clear understanding of what restrictions must be placed on g and f for such processes to be accepted as "biology-like". For biology, f is restricted to physical processes. For evolutionary algorithms, f may accept any process that can be described as a computation. Map g is always stochastic. It is argued in this

1/21/05

paper that only when g belongs to a particular class of phenomena, named here the "iterated probabilistic computations with selection" (IPCS) and when selection is based on competition or interaction among multiple evolving entities, is it appropriate to see a process as evolutionary in the biological sense. With this definition, biology-like evolution can always be described as a computation. A computational definition of evolution allows for quantitative approaches to traditionally difficult issues such as biological complexity, evolutionary progress, and evolutionary innovation.

Continuous versus discontinuous phenomena

Notions taken from computational complexity theory are often invoked to explain real-world phenomena without explicitly acknowledging that the formal theory applies only to discrete phenomena. Modern physics, in spite of quantum mechanics, assumes that space and time are continuous. Perhaps one of the greatest questions facing modern science is whether or not this assumption is true (Smolin, 1997; Wolfram, 1983, 2002). If the answer is no, then physical law as customarily presented in the form of differential equations is only an approximation of reality and computation provides a better formalism for understanding our universe (Wolfram, 2002). This grand question of physics, though interesting, is not central to this paper. The argument made here is that the informational nature of DNA is not continuous (even if the universe is), and that the evolution of DNA may be properly treated as a computation. It follows from this that computational notions of complexity may be formally used to characterize DNA sequences and that the evolution of DNA can be properly treated as a probabilistic computation even if the evolution of tissues, organisms and ecosystems may not.

1/21/05

Computation

The mathematics of computation applies to processes characterized by having information containing [discrete] input that is acted upon according to a defined set of [discrete] rules producing an information containing [discrete] output. In other words, processes that can be diagramed as:

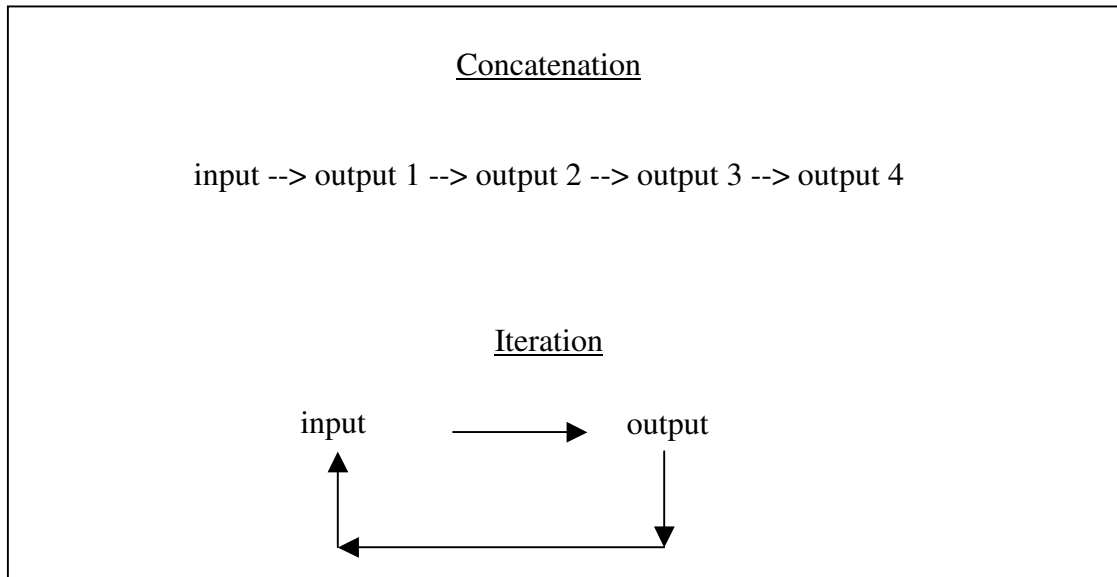
**input (information) \Rightarrow manipulation of that information according to
rules (computation) \Rightarrow output (information).**

This class of maps is very large and is called partial recursive when not all inputs lead to defined output (Li and Vitanyi, 1997). The computation itself (implementation of the rules) is formally considered to be carried out by a physical device that imitates what a trained human being might do. It is important to distinguish two classes of computation, deterministic computation and probabilistic computation. Deterministic computations are familiar and obviously useful. The same input always produces the same output, a very desirable property when one is interested in answers to clearly stated problems. Probabilistic computations are less familiar because they may produce different results each time they are performed. In a probabilistic computation, one or more of the manipulation rules employs a stochastic action. Probabilistic computations play an important role in computational theory and are useful for describing some natural phenomena.

Computations, by definition, either produce output and stop, or are not defined. Many real-world phenomena are on going. One way to bridge this gap is to describe on-going phenomena in terms of concatenated or iterated computation. Such computations are diagramed in Figure 2, and are characterized by the output of one computation serving as the input to the next. Chains of computations continue on through time until there is no output, and regularly produce intermediate outputs which represent the current state of the system.

Figure 2

Schematic diagram of concatenated and iterated computations.

**Information**

Formally, computation theory deals with situations where information is encoded in a linear sequence of symbols. For convenience, the preferred symbols are "0" and "1" (binary code). The theory also applies to higher dimensional encodings so long as the encoding can be reversibly and straightforwardly reduced to a linear sequence of symbols.

In 1948, Claude Shannon (Shannon and Weaver, 1949) proposed a quantitative definition of information based on probabilities. The underlying idea is that when probabilities can be assigned to all possible outcomes presented by a situation, then something must be "known" (or determined) about that situation (Jaynes, 1978). Since information is additive and probabilities are not, Shannon proposed the logarithm of the probability as a measure that corresponds to our

1/21/05

everyday notion of what is known or what is determined. His formal definition is given by the relationship:

$$\mathbf{H} = -\sum p_i \log_2 p_i. \quad (1)$$

Where the summation is over "i" possible states, and p_i is the probability of the i^{th} state. Shannon formally referred to H as entropy, but he and many others have used H to measure information. H is given units of bits. Equation 1 defines a self-consistent additive measure of the freedom of choice present in any discrete probability distribution that characterizes the possible states or outcomes available to a system.

The definition of H is quite general and allows calculation of information (or entropy) in any situation where a discrete set of possibilities and their associated probabilities can be identified. Because of this, nearly all arguments about the "information present" amount to discussions of the proper number of states and corresponding probabilities that apply in a particular situation. For example, when discussing the "information content" of a binary string of length n, it is common to make the assumption that the string was drawn with equal probability from the set of 2^n possible strings of length n. When this assumption is made, equation 1 reduces to:

$$\mathbf{H} = \mathbf{n}. \quad (2)$$

Because of this, it is sometimes stated that the "information content" of a binary string is given by the length of the string. When discussing information in the context of computations, however, it is clear that 2^n may not always be the number of states of greatest interest, and in many situations the uniform distribution does not properly reflect what is known (or determined) about the probabilities of those states.

1/21/05

Complexity

Direct application of equation 2 can create impossible dilemmas. For example, the length of the input to a computation is often not the same as the length of the output. Sometimes it is possible to design a second computation that will recreate the original input from the original output. In these cases it is evident that no information is lost during the computational cycle. Accepting equation 2 as a primary definition of information would force us to conclude that when the input is not the length of the output, information is lost and magically recreated in such a cycle.

Another problem is easily appreciated by comparing the information content of a string consisting of 100,000 zeros and a string of the same length encoding an entry from the Encyclopedia Britannica. Equation 2 yields the same amount of information for both strings.

Both of these examples violate our common sense notions of information. Computational theory resolves these dilemmas by defining differently the appropriate number of states to be used when applying equation 1.

The algorithmic information content, or Kolmogorov complexity, of a binary string x with length n , is informally defined as the length of the shortest binary input to a universal [Turing] machine that results in output x [exactly]. This quantity is usually denoted as $K(x)$ and is consistent with equations 1 and 2 as long as we understand that the number of states is $2^{K(x)}$, not 2^n . $K(x)$ is a fundamental concept in computer science and provides a widely accepted definition of both information and complexity. $K(x)$ has the property that it is minimal for a string of all zeros (or all ones), and greatest (and approximately equal to n) for a long string of "randomly" ordered zeros and ones. $K(x)$ depends somewhat on the universal machine used, but has the property that

1/21/05

$K(x)$ for one machine differs at most by a constant independent of x from the $K(x)$ for another machine. In the limit of very long strings, the machine dependence is insignificant.

By definition, the $K(x)$ value of the output of a deterministic computation cannot be greater than the $K(x)$ value of the input. The $K(x)$ value of the output can, however, be less than that of the input (in such a case, the input cannot be recreated deterministically from the output because "information" is lost during the computation).

$R(x) = n(x) - K(x)$, where $n(x)$ is the length of string x , defines a derived information measure variously known as the compressibility or redundancy of x . $R(x)$ measures how much longer string x is than the shortest possible input that could output x on some universal machine. It is useful to think of this shortest possible input as the most compact (or compressed) representation of string x , and it is a provable fact that for very long strings, those with $K(x)$ values at least equal to their length, are incompressible and contain no internal redundancy.

Depth

Interestingly, $n(x)$, $K(x)$ and $R(x)$ are not sufficient to fully characterize the complexity of a binary string. The time required to compute various equal length strings from their shortest representations differs greatly depending on the particular sequence of zeros and ones. Logical depth (also called computational depth) was introduced by Charles Bennett (1988) and refined by Juedes et al. (1994) to capture this property of strings. It is informally defined as the minimum number of elemental steps required for a universal [Turing] machine to produce output x from nearly-incompressible input.

1/21/05

Logical depth provides a formal definition for the amount of computation required to produce an object from primitive initial input (a nearly random string). It encompasses an aspect of complexity that is not measured by $K(x)$. Logical depth has the property that it can be created or lost during a deterministic computation, but creation of depth is subject to a slow-growth law (Bennett 1988, Juedes et al. 1994). The slow-growth requirement is that deep output can only be created rapidly from deep input. Creation of deep output from shallow input requires extensive computation.

Logical depth measures a minimum time investment inherent to a digital object. The standard definition is made in terms of elemental computation steps. A slightly different definition allows the time investment to be measured in bits. This definition relies on the observation that every elemental computation step can be described (or encoded) in a small number of bits. Because of this, a computation can be recorded as a binary string. This string is sometimes called the "history" of the computation. It is an exact transcript of all the steps involved in the computation. We assume that there exists a universal optimal encoding, and that this encoding is used. When defined in this way, the computation itself becomes a binary object and the length of this object provides measure of the amount of computation required to produce its output. This device allows information-like measure (compatible with equations 1 and 2) of computations as well as their inputs and outputs.

If one considers only computations that output a particular string x [exactly] from nearly-incompressible input, there must be some computation whose length (history) is either equal to

1/21/05

or shorter than any other. The length of this shortest computation starting from nearly-incompressible input measured in bits characterizes the same fundamental property of a digital object as does logical depth (measured in elemental steps). We will designate the length of this shortest history measured in bits to be $D(x)$, and in those situations where we need to distinguish it from logical depth, we will refer to it as transcript depth. $D(x)$ characterizes the shortest possible computation that outputs x [exactly] starting with input that has very little internal organization. $D(x)$ is also seen to measure the minimal difficulty of creating x from primitive beginnings.

Defining depth in bits allows depth to be related directly to other information and complexity measures. Information is additive, so it is legitimate to combine various information measures when there is a justifiable reason for doing so. The history of any computation must include the reading of the input and the writing of the output. So, letting the length of the nearly-incompressible input that initiates the shortest computation that outputs x be $L(x) = K(x) + s(x)$, where $s(x)$ is small, we can write:

$$\mathbf{D(x) = L(x) + n(x) + W(x).} \quad \mathbf{(3)}$$

This relationship defines a new measure, $W(x)$, that we may think of as “path information” or “computational effort.” $W(x)$ characterizes the minimum amount of manipulation required to produce output x (of length n) from nearly-incompressible input of length $L(x)$. Introducing the definition of redundancy leads to:

$$\mathbf{D(x) = 2K(x) + s(x) + R(x) + W(x).} \quad \mathbf{(4)}$$

$K(x)$ appears twice, because every computation must include reading of the input and writing of the output. $R(x)$, $s(x)$ and $W(x)$ measure aspects of x that are not captured by $K(x)$.

1/21/05

Through these exercises, it is argued that $D(x)$ provides a more comprehensive measure of the complexity of binary string x than traditional measures. It incorporates aspects of complexity not usually recognized. When employing this measure of complexity, redundant objects have greater complexity than non-redundant objects of the same length. For redundant objects, $R(x)$ and $W(x)$ can be much larger than $K(x)$ and $L(x)$, $R(x)$ is always bounded above by $n(x)$, while $W(x)$ can be very large.

To gain a better understanding of $D(x)$, it is useful to consider various verbal descriptions of depth that have been used by others. Bennett (1988) likened logical depth to the "value" inherent in an object, and he tied value directly to the "amount of computational history evident in an object." Juedes et al (1994) refer to "computational usefulness", Lutz (personal communication) to the "organization of redundancy", and Lathrop (1997), referring back to Bennett and to Juedes et al, variously saw depth to be manifest as "intricate structure", "computational work stored in the organization of x " or to "organization embedded in a string by computation". Thus, we see that Bennett's use of value is based on the notion that information present in input can be organized by the process of computation into output structure that exhibits increased usefulness for further computation. Such organization is evident in the structure of some binary strings, and when detected, the most plausible explanation is that the string was the product of extensive computation (i.e. that it has a long computational history).

This fundamental property of digital objects can be technically defined in various ways. For convenience I will use the generic term "depth," recognizing that the current mathematical

1/21/05

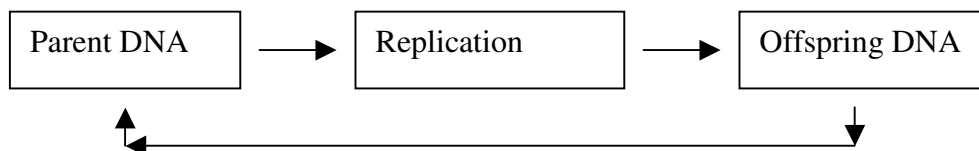
literature primarily deals with the specific definition referred to here as logical (or computational) depth. The literature makes clear that the potential for significant depth is minimal for objects that have no redundancy and also for objects that have very high redundancy. Significant depth is only possible (but not required) when intermediate values of redundancy are present.

Evolution as Computation

Modern molecular genetics provides abundant evidence that information encoded in DNA nucleotide sequences plays a central role in life processes. Because DNA information is encoded in discrete form and because all sequence changes are discrete, it must be possible to describe the changing of DNA sequences over time in computational terms. Because DNA sequence change mechanisms are at least partly stochastic, the appropriate class of computation must be probabilistic. And, since old DNA sequences provide the starting point for new DNA sequences, the appropriate model must be iterative. In broad terms, the computation we are seeking must be like that diagramed in Figure 3, where the output of one round of computation serves as the input to the next.

Figure 3

DNA evolution as a computation.



Over the past 15 years or so, a sub-field of computer science has arisen that is devoted to the development and use of "genetic" or "evolutionary" algorithms. In general, these algorithms

implement various concepts borrowed from biological genetics and biological evolution (Goldberg 1989, Holland 1992). The details often look quite different, but always there is (1) an encoded body of information that specifies a behavior or a structure, (2) a mechanism for generating multiple "offspring" information bodies incorporating quasi-randomly generated changes, and (3) selection of the "best" offspring according to specified criteria. The selection is often based on behaviors or structures deterministically generated from the information bodies. The chosen offspring then serve as the "parents" for the next iteration (generation). When properly designed, such algorithms are frequently able to find good solutions to otherwise intractable problems (Goldberg 1989; Miettinen, et al, 1999). Fogel (1994) formally described such processes in terms of a difference equation:

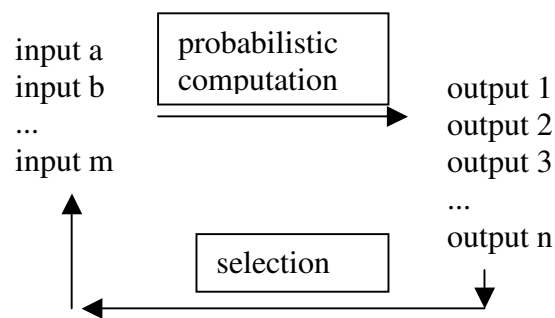
$$x[t + 1] = s(v([t])), \quad (5)$$

where x is the evolving object, t is time, v is random variation, and s is selection.

They may also be defined in terms of a diagram. A diagram that incorporates the three defining characteristics of the sought-for class of algorithms is illustrated in figure 4.

Figure 4

Iterated probabilistic computation with selection (IPCS)



Where $m < n$

1/21/05

We will refer to this class of computations as the "iterated probabilistic computations with selection" (IPCS). In the diagram, the box labeled "probabilistic computation" is understood to include a stochastic process, and selection is based on concisely defined criteria. Frequently, a number of such computations are carried out simultaneously and the outputs interact through a recombination-mimicking mechanism. A characteristic feature of IPCS is that stochastic input in combination with non-random selection is capable of generating increasingly non-random inputs. IPCS systems can generate increasing $K(x)$, $n(x)$, and/or $D(x)$ depending on the selection criteria. Because random numbers are shallow, the process can only accumulate depth slowly.

As a class, these computations exhibit a variety of phenomena familiar to population and evolutionary geneticists. Among these are error catastrophe, stasis, and innovation. From experience, we know that, computer algorithms belonging to the IPCS class can generate outputs that get better and better at predetermined tasks over time. This is true for both simple and complicated tasks.

The class of processes encompassed by the diagram in figure 4 would seem also to include the evolution of DNA. The implication of this observation is far reaching. It suggests that modern organisms are based on four billion years of iterative computation. That computation lies at the heart of the evolutionary process will not be easily accepted by some, and deserves careful scrutiny. Several concerns need to be specifically addressed. First, DNA does not consist of abstract symbols, but is a physical molecule. Second, DNA is an integral part of a living organism and as such participates in continuous rather than discrete processes. Third, natural

1/21/05

selection does not operate directly on DNA but at the level of organisms. Fourth, we must be certain that the traditional mathematical notion of computation is valid when the computer itself changes as a consequence of the evolutionary process. Fifth, the evolutionary change mechanisms, mutation and recombination, are not strictly random.

The first concern (DNA is not abstract) should not present a difficulty. The mathematical theory of computation employs physical machines with one or more physical tapes. When a desktop computer reads instructions recorded on its hard drive, the yes/no symbols that encode the instructions consist of molecular magnetic domains. With DNA, the encoding system is quaternary (base four) and each of the four symbols is represented by a nucleotide-pair that is a molecular unit of a very long linear molecule made of many such units. The chemical form of the representation is a detail that only becomes important when one wishes to understand the machine that reads the message. The abstract notion that information is encoded in a language of four symbols is independent of the recording medium. There is no theoretical barrier to prevent the building of a computer that uses DNA molecules instead of magnetic tapes.

The second concern (the continuous nature of life) is best addressed by examining how DNA functions in the cell. DNA functions in two basic ways. One is by providing a three dimensional chemical shape that other molecules bind to by means of non-covalent interactions. Of particular interest are protein molecules that bind to particular sequences of nucleotide-pairs within the linear structure of DNA. These proteins "read" specific nucleotide sequences by binding to them. Two states are possible, bound and not bound. Because molecules bind and unbind at a rapid rate, continuous outputs are achieved by averaging the fraction of time that a molecule is

1/21/05

bound. The underlying signal is digital, on or off. The second way that DNA functions is by being copied. Much of the information encoded in DNA is expressed by first being copied in the form of smaller RNA molecules. RNA molecules encode information in a very similar four-symbol language, and an RNA molecule carries the replica of a small segment of the information present in a DNA molecule. This information is used in three ways. First, RNA molecules fold up into complex three-dimensional shapes that depend on the linear sequences of nucleotides that constitute the RNA molecule. This folding is one aspect of the genotype-phenotype map (map *f* in figure 1). Second, proteins bind to specific nucleotide sequences present in RNA molecules much like the proteins that bind to specific DNA sequences. Thirdly, for many RNA molecules, the sequence of nucleotides that constitutes an RNA molecule specifies in a one-to-one manner the sequence of amino acids that constitutes a protein molecule. In this "translation", the RNA nucleotides are read linearly three at a time to specify the order of 20 amino acids that constitute protein molecules. In this process, a base four language is translated into a base twenty language. Because the three letter code is redundant (there are 64 three letter nucleotide codes) information is lost in this process. The translation is purely deterministic (except for errors). The information present in the 20 letter amino acid language is used in the cell to determine the three dimensional shape of protein molecules. Thus, like RNA molecules, protein molecules fold up into complex shapes which are determined by the internal sequence of units, in this case amino acids. For both RNA sequences and protein sequences, there are many-to-many relationships between particular sequences and particular shapes. Presumably nature goes to the trouble of translating from the four letter nucleotide code to the twenty letter amino acid code because the repertoire of chemical shapes available to protein molecules is richer than that available to RNA molecules. This information-based folding of macromolecules is central to life

1/21/05

as we know it but is not relevant to the question of whether or not the evolution of DNA is a computation. At the core, information encoded in the linear order of nucleotides in DNA molecules is read out in two ways, it serves as the basis for protein binding (yes/no signal) or it is copied. Thus, we conclude that the informational functions of DNA are discrete.

The third concern (selection at the level of organisms) does not present an insurmountable problem either, provided that there is a clear (substantial) link between the organism level and the DNA level. In biology, this is obviously true because DNA molecules are an integral part of the organism. When an organism dies, so does the DNA, when an organism reproduces, the DNA is replicated. Thus, even though selection operates at the organism level, there is a direct causative linkage between organism selection and DNA selection.

The fourth concern (encoding of the machine) is fundamental. It requires examination of how the computer is defined in computer science. Turing envisioned a mechanical (unchanging) device that would read and write to one or more tapes according to deterministic rules encoded in the input plus a small amount of information necessary to initialize the system. So, the question is, does having the computer itself encoded in the input change the relationship between the computer and the input and output in a way that invalidates the mathematics of computation? The existence of universal machines would seem to answer this concern. A universal Turing machine can be programmed to carry out the actions of any other Turing machine. The input algorithm adds whatever information is necessary to define the specific machine that is required to carry out the desired calculation. Employing an evolving algorithm (and machine) does not change this relationship.

1/21/05

The fifth concern (the non-randomness of variation and mutation mechanisms) is not of great concern since the theory requires only that there be a stochastic aspect to the change mechanisms. In biology, the mechanisms are biased, but hardly deterministic.

If one accepts that the evolution of DNA is indeed a computation, there remains the question of whether or not all processes that can properly be described as IPCS evolve over time in a "biology-like" manner. Selection in biological evolution is a function of the environment and the environment of most living organisms is defined in part by other evolving organisms. An IPCS operating in an environment of fixed-rules seems rather sterile compared to what happens in the world of biology. Engineers refer to fixed-rule activities as optimizations, and optimization does not characterize what happens (over the long term) in biology-like evolution. When evolutionary computer algorithms are designed such that selection is based on competition among multiple evolving entities, the resultant outcomes seem to be more "interesting" and more "biology-like" than when the environment is unchanging (Axelrod, 1984; Miller, 1989; Stanley, et al, 1994). Thus, an attractive possibility is to restrict the definition of biology-like evolution to the subclass of IPCS for which selection is based [in part] on competition among multiple evolving entities.

General Definition of Genome and Phenotype

Before this tentative definition can be examined more fully, it is necessary to have general definitions for genome and phenotype. For life on Earth, the definition of genome (total DNA sequence counted once) and the concept of genome space (the hyperdimensional space of all DNA sequences) are well established. It is natural in this paper to make the definition in terms

1/21/05

of binary strings and structures that can be reversibly and straightforwardly reduced to binary strings. The proposed definition comes directly from the notion of computation, which requires a discrete (finite) body of information to serve as the input to a computation. Generally, the body of information is in the form of a finite sequence of zero's and one's (a binary string). It seems natural to define genome to be the input/output that characterizes an IPCS at a particular time. Phenotype is simply any expression or consequence of the input/output that provides a basis for selection. In the simplest cases, the genotype is also the phenotype.

IPCS With Selection Based On Competition Among Evolving Entities

An apparent characteristic of biology-like evolution is the generation of complexity over long periods of time. Because of this, any quantitative theory of evolution that suggests a natural measure of the complexity generated should have appeal. McShea (1994; 1996) reviewed numerous attempts to characterize complexity increases over time in biological systems using traditional complexity measures. These efforts suggested either that a new concept of complexity was needed or that complexity increase may not be a fundamental aspect of evolution. Viewing evolution as a computation sheds new light on this issue.

Several measures of complexity were introduced earlier. These include algorithmic complexity $K(x)$, length $n(x)$ and depth $D(x)$. It is known that $K(x)$ and $n(x)$ do not fully characterize a binary string (Bennett, 1988). The missing part is the amount of computational time or effort that must be invested to organize information into a useful form. $D(x)$ provides measure of this minimal computational investment. Intuitively, a measure of complexity should be negentropic in the sense that effort is required to increase it. $K(x)$ is actually entropic in the sense that if

1/21/05

random changes are introduced into a nonrandom (redundant) string of fixed length, the $K(x)$ value of the string is almost certain to increase because most bit strings are incompressible (Li and Vitanyi, 1987). $n(x)$ is negentropic in the sense that thermodynamic work must be done to increase the length of a string, but $n(x)$ does not fit our intuitive notions of complexity since it does not distinguish simple from a complicated strings of the same length.

$D(x)$ measures of the minimal computational effort required to create something non-random from something random. Computational effort as used here should not be confused with thermodynamic work. As Bennett showed, there is no lower limit to the amount of thermodynamic work required to carry out a computation provided that no bits are erased and the output is not longer than the input (Bennett, 1982). Thus, the $W(x)$ part of $D(x)$ (equation 3) can be created by an ideal computer without thermodynamic work being performed. Computational effort reflects the time (measured in steps) required for a particular manipulation rather than the energy required. The central paradigm underlying computational theory in which actions are carried out in discrete steps, each requiring finite time, establishes the equivalence of time and activity. Because actions can be encoded digitally, the time required for the action can be measured in bits. This, in turn, allows the definition of entropy like measures that incorporate computational time. $D(x)$ is negentropic in the general sense of requiring computational effort, but not in the thermodynamic sense of requiring investment of energy.

Since IPCS's are computations, they are capable of generating depth. A question is, do they have any special propensity to do so? It is easy to find examples of IPCS based computations that do not generate depth. For example, imagine a simple evolutionary algorithm that does not allow

1/21/05

any change in the length of the output and for which selection favors zeros. Iteration of this algorithm will quickly converge to strings of all zeros. This IPCS based computation minimizes both $K(x)$ and $D(x)$. Selection criteria play a central role in determining all trends in IPCS based systems. Therefore, we must ask if there are special selection conditions that favor increasing depth.

If depth is favored in particular evolutionary systems, then depth must be favored by selection. In biological systems this means that depth must offer a fundamental advantage. The slow-growth law of depth generation provides a rationale for why this might be so. In nature, selection is strictly a matter of survival and reproduction. To live and reproduce, organisms face innumerable time-dependent challenges. If a challenge and corresponding response are treated as computations, then a deep response is one that requires extensive computation to produce. The nature of depth allows much of the necessary computation to be performed in advance and stored in the form of [deep] structure. This structure yields quick deep responses in predetermined ways. A shallow entity cannot generate a deep response quickly -- clearly a problem in many situations. Quick shallow (either random or simple) responses are always possible; so if deep structures are favored, quick deep responses (responses based on extensive computation) must provide advantage. It is easy to imagine that animal brains and elaborate metabolic networks are examples of deep structures that have been selected because they provide rapid effective responses to complicated situations. Unfortunately, we do not know how to define the depth of physical structures such as brains, metabolic networks, and organisms. Depth is formally defined for binary objects or other structures that can be simply reduced to binary objects. DNA is such a structure, but an organism is not.

1/21/05

But what sorts of challenges require deep responses? Computation is of little use if a challenge is random, and only minimal computation is required if a challenge is easily predictable.

Assuming that every challenge can be represented as a binary object, and that all binary objects represent potential challenges, then the vast majority of potential challenges are random. While a few simple challenges are more effectively met by a response based on extensive computation, it must be that most challenges that are better met by a deep response are themselves the product of extensive computation. In other words, there is a tendency for deep challenges to demand deep responses. But how are deep challenges created? In the world of biology, the physical environment an organism experiences is probably not very deep. Most aspects of the physical environment can be characterized as either random or repetitive in a simple way. Thus, if organisms needed only to contend with the physical environment, we should not expect them to evolve deep structure. However, when the environment contains other evolving agents the situation is fundamentally different. In this case, each entity experiences challenges created by other agents. Typically, when a response is made to such a challenge, the response itself creates a challenge for other agents. Thus, an endless challenge-response-challenge cycle is established. This cycle differs fundamentally from the noncyclic challenge-response relationship with the physical environment. As evolving agents challenge and respond to one another, selection will always favor more effective responses and penalize those that do not respond. This is the “Red Queen” principle first articulated by Van Valen (Van Valen, 1973). Since an important aspect of effectiveness is timeliness, whenever more effective responses are made possible by extensive computation and are time-dependent, such responses will tend, over time, to become based on deeper structures. But deeper responses create deeper challenges for other agents, so a positive

1/21/05

feedback loop is established that favors the generation of depth. This loop is expected to form in nonliving as well as living IPCS systems that feature competing agents. Thus, a clear prediction of the computational theory of evolution is that complexity as measured by depth should increase in evolutionary systems that feature interaction between evolving agents.

This positive feedback loop is similar to positive feedback loops proposed by others for more traditional measures of complexity (Edmonds, 1999; Heylighen, 1999), but the argument is stronger for depth because of the importance of time. The slow-growth law places a fundamental limitation on the minimal time required to generate anything that is deep unless the input to the process is itself deep. Given that many responses to external stimuli need to be made quickly, it is inevitable that preexisting deep structure will provide advantage in some situations. In a system of competing agents, advantage for one agent creates problems for others. Thus, useful depth gained by one agent will create selective pressure for the acquisition of depth by others. The argument based on traditional complexity measures is similar, but without a slow-growth law. In other words, complexity gained by one agent will create problems for other agents that results in selective pressure to acquire additional complexity. The problem with this argument is that it is difficult to make a direct link between the complexity of an agent and the complexity of the problems it creates for other agents. The best attempt to do this is based on control theory (Heylighen, 1999). Ashby's Law of Requisite Variety states that "In order to achieve control, the variety of actions a control system is able to execute must be at least as great as the variety of environmental perturbations that need to be compensated" (Ashby, 1956). Phrased in the terminology of complexity, Ashby's law says that in order to assure internal stability, an agent must be at least as complex as the environment it contends with. Regardless of how complexity

1/21/05

is measured, this is impossible to achieve [for all agents] in a system which consists of competing agents; but this imbalance provides continual selective pressure for the agents to increase in complexity. Two factors mitigate this selection. The first is maintenance cost. Complexity must be maintained and the greater the complexity the greater the cost. Eventually cost cannot exceed the benefit. The second is time. If an agent is organized as a table or database of potential responses corresponding to possible perturbations, it must take longer to find an appropriate response as the table becomes larger. Thus, as an agent becomes more and more complex, it becomes more and more unable to respond to any particular perturbation in a timely manner. The control theory response to this limitation is the Law of Requisite Hierarchy (Aulin, 1979), which states, "The larger the variety of possible responses, the larger the control hierarchy needed." But control hierarchy is particular kind of organization, and organization is what is measured by depth. So, the control theory argument also leads to the expectation that depth should increase in evolving systems characterized by competition between evolving agents. What computational theory makes clear, but control theory does not, is that the organizational overhead can become much larger than the complexity (i.e. $K(x)$) being organized.

The basic prediction that evolutionary systems featuring competing agents are expected to exhibit increasing depth over time is testable. A favorite evolutionary system for studying competition is iterated prisoners dilemma in which the players are evolved by means of a genetic algorithm (Axelrod, 1987; Stanley, et al, 1994). Selection is based on one-on-one play between multiple players. Typically, scores are averaged over many rounds of play and selection is based on a player's average score. When the players are placed on a two-dimensional board and

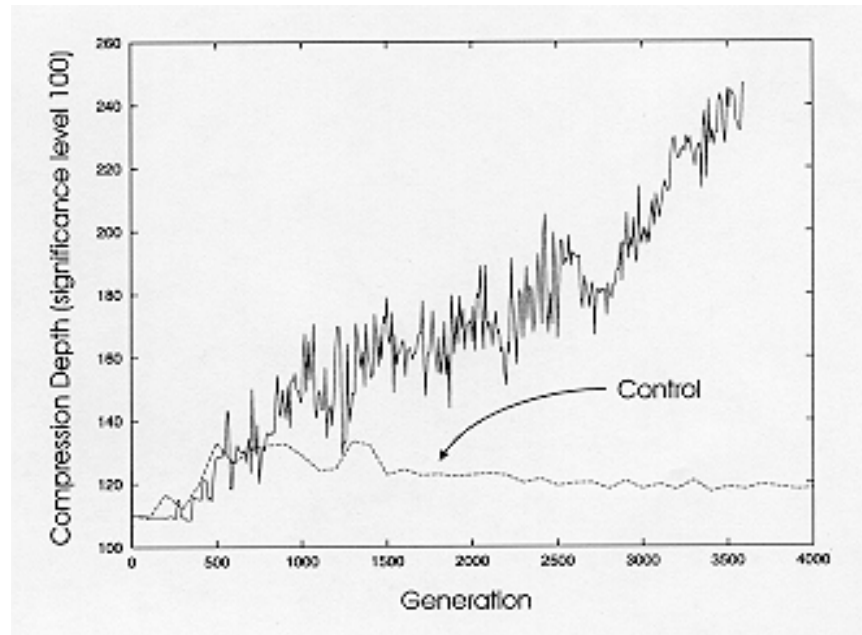
1/21/05

players only play their neighbors, a remarkable thing happens. After the initial phase during which inept players are weeded out, absolute scores do not generally improve, but players continue to get better at playing the game. This can be demonstrated by competing more evolved populations of players against less evolved populations. When this is done, the more evolved almost always dominate the less evolved (Ashlock and Mayfield, 1998). Getting better at playing prisoners dilemma means that players get better at responding to pattern in an opponent's play; they develop increasingly sophisticated offensive and defensive strategies. In the experiments performed by Ashlock and Mayfield (1998), the maximum number of states available to each finite state player was fixed. This limitation means that agents could not increase the total amount of "information" available to them, and implies that agents used information more efficiently or more cleverly as the system evolved. This outcome is consistent with the hypothesis.

A more direct experiment was reported in 1997. Lathrop (1997) defined and characterized a computable depth-like measure, compression depth, based on data compression. He showed that any lossless parametrizable compression algorithm can be used to define a specific measure of depth. Lathrop wrote an algorithm that measured compression depth based on the Lempel-Ziv (LZ) compression algorithm (Lempel and Ziv, 1978). He called this measure, LZ compression depth, and showed that for certain cellular automata (unpublished) and for a system in which finite state machines played iterated prisoner's dilemma and were evolved by means of a genetic algorithm, the LZ-compression depth of the state structures steadily increased over time (Lathrop, 1997; Lathrop, 1998). An example taken from his Ph.D. thesis, with permission, is shown in figure 5.

Figure 5

Accumulation of L-Z Compression Depth during an Adaptive Process (Lathrop, 1997)



Compression depth measures organized redundancy that can be found by a particular compression algorithm. The more computational resources needed to uncover the redundancy (and thus, the more complicated the redundancy), the greater the compression depth. Clearly compression depth is related to logical and transcript depth. One problem is that no known compression algorithm is able to find all possible redundancy if limited by finite computational resources (time and memory). Wolfram (2002) shows convincingly that there exist many simple algorithms that are capable of generating essentially limitless complex redundancy and for which no known compression algorithm is capable of finding that redundancy (in reasonable times). In other words there exist binary strings that are random by any known test, yet are known to be algorithmically simple (have short programs). Some of these strings are also deep. Less clear is whether or not there exist strings that are compression deep, but logical (or transcript) shallow. If so, they are extraordinarily uncommon.

1/21/05

Any evolutionary theory which predicts an increase in complexity over time must be able to explain the existence of lineages that become simpler over time and also lineages that appear to remain at the same level of complexity for long periods of time. The twin issues of evolutionary regression and evolutionary stasis have been addressed by others from two quite different angles, environmental complexity and opportunity. The introduction of a computational theory of evolution does not change these arguments but adds a third. Namely that previous analyses did not employ a proper measure of evolutionary complexity. The clearest examples of evolutionary lineages that have apparently decreased in complexity occur when a parasitic life style is adopted. One obvious consequence of such a lifestyle change is reduction of the number of other organisms that contribute to the environment. An environment consisting of a single organism must be simpler than an environment consisting of many interacting organisms. So, given the metabolic cost of maintaining a complex organism, it seems reasonable that simpler organisms would be the natural consequence of change to a simpler environment. This argument holds for any reasonable measure of complexity.

Opportunity has been much discussed (Gould and Vrba, 1982; Nitecki, 1990; Stadler et al, 2001). In biology, organisms are selected, but mutations occur in DNA. When a new DNA sequence is produced in the germ line, it is converted to an organism according to the f map. It may be that for a particular lineage, and a particular environment there are no easy (highly probable) DNA changes that yield a better organism. In such cases, evolution will produce no significant changes over long periods of time. Such situations should be more common in stable environments populated by other organisms that are also experiencing little opportunity for

1/21/05

change. One can imagine that only on rare occasions some highly improbable change will lead to a breakthrough and a subsequent cascade of adjustments, followed again by stasis. It may also be that some lineages become so stable that evolutionary breakthroughs almost never occur unless enabled by catastrophic external changes in the environment (Raup, 1994). What is not known is whether or not apparently static lineages are accumulating DNA sequence complexity that is not easily recognized as structural complexity. This idea seems plausible, but is not easily testable for living systems. Evolutionary algorithms provide rich opportunity to test such ideas experimentally.

Other Features of Evolution Better Understood In Terms of Iterated Computation

Any useful new theory should naturally explain observations that existing theories have difficulty with. Modern evolutionary theory does not provide full explanation of several well-known observations. Generation of complexity over time is one of these; two others are increasing diversity over time and innovation.

A frequently noted characteristic of the evolution of life is a trend toward increasing number and diversity of life forms over long periods of time. Some authors have seen increasing diversity as so central to the evolutionary process that they proposed increasing diversity to be the engine that drives the evolutionary process (Brooks and Wiley 1988). This amounts to proposing that evolution is an entropic process. In light of this suggestion, it is interesting to ask whether or not analysis of the evolutionary process as an IPCS indicates any fundamental requirement for increasing diversity. Even a limited knowledge of genetic algorithms and their outputs reveals that there is no such requirement for algorithmic evolution. Most evolutionary algorithms

1/21/05

generate decreasing diversity over time. It is also common to restrict diversity to economize on computer memory. Clearly, increasing diversity is not central to IPCS based computation.

Adding the requirement that selection be based on competition among evolving entities does not change this conclusion. The key parameters governing diversity trends are well recognized by evolutionary biologists. When selection is sufficiently stringent, a single genotype is the expected output of an IPCS based system even if the system is initially seeded with many genotypes and regardless of the basis for selection. This is also the expectation for natural systems (Hartl, 1988). Careful reflection on IPCS with competition should convince one that the generation of diversity during biology-like evolution depends on system specific details.

Diversity trends result from opposing forces. Stochastic mechanisms generate diversity and selection decreases it. The net result can either be increasing or decreasing diversity, depending on the balance. Thus, the observed increase in life-form diversity over time does not stem from any fundamental requirement of the evolutionary (IPCS) process, but derives from the particular rules in play in the biosphere. These rules are broadly understood by evolutionary ecologists, and represent an active area of continued investigation.

Much has been written about how the current evolutionary paradigm does not fully explain major innovations in evolutionary lineages (Behe, 1996; Holland, 1998; Kauffman, 1995). Most biologists accept that incremental change is compatible with innovation but may not fully explain it, while a few claim that incremental change is incompatible with major evolutionary innovation and that some new principle is needed (Behe, 1996). Whatever position one takes, it is clear that current biological theory does not provide a mathematical understanding of innovation.

1/21/05

There are two widely published concepts that attempt to address this issue. One is variously referred to as co-optation or preadaptation, or more generally as "exaptation" (Gould, 1988; Gould and Vrba, 1982), and the other is "emergence" (Holland, 1998). Exaptation is a term invented to describe situations where organismal features were evolutionarily derived from earlier features with very different structures or purposes. Thus, fish fins set the stage for walking on land, swim bladders for breathing air, and signal transduction mechanisms preceded vision. Emergence describes those situations in which a [deterministic] process leads to an outcome that could not have easily been predicted by [human] observation of the initial state. The development of an adult human being from a fertilized egg cell provides a good example.

The science of computation is rich with similar phenomena. Emergence happens every time a complicated computation is executed. This is why it is so important to document a long computer program. The phenomenon is most obvious when there are chains of computations where the output of one computation (or sub routine) serves as the input to the next, but examples are also known of simple algorithms (Wolfram, 2002). In many cases it is quite impossible to determine the final output by examination of the initial input (except by actually performing the computation). By definition, the emergence phenomenon becomes more pronounced the less the output resembles the input. At the extremes, a computation that outputs x from input x (the copy function) exhibits no emergence while any computation that produces output through a long nontrivial process from compact input will exhibit lots of emergence. By definition, a random (maximally compressed) input to a computation reveals nothing about the eventual output (except its algorithmic complexity). In general, the more computation that is

1/21/05

required to produce a particular output the less obvious is the output from the input. Thus, the nonquantitative notion of emergence is closely tied to the quantitative notion of depth. In fact, depth offers a possible quantitative measure of emergence when the term is applied to computation.

Computational theory provides a possible foundation for the notion of exaptation. One feature of deep binary objects is that neighbors of deep objects in [hyper-dimensional] binary sequence space are also deep. This is a direct consequence of the slow-growth condition (deep objects can only be quickly computed from other deep [i.e. near by] objects). Whenever objects that are near by in sequence space can serve as inputs to computations that output objects with different properties, the underlying condition for exaptation is established. That such situations are common is fundamental to the usefulness of computation. If it were not so, computer programming experience gained in one situation could not be usefully applied to other situations. The two track trajectory presented in figure 1 (Stadler et al, 2001) was invented to analyze and illustrate this phenomenon. Growth and development ($\text{map } f$) serves to further amplify the underlying opportunities presented by the fundamental nature of computation. Iterative selection is capable of identifying and amplifying small initial differences. Thus, seeing evolution as a computation demystifies innovation and demonstrates that there is no fundamental problem associated with it.

Self-organization

Much has been written about "self-organization" of physical systems, yet details about how self-organization comes about remain vague. There exist some theoretical works that suggest self-

1/21/05

organization may be favored "near the edge of chaos" (Lewin 1992, Kauffman 1993, Kauffman 1995), but this broad concept tells us little about mechanisms. Nor do these theories tell us how organization should be defined or measured. Depth provides a formal comprehensive measure of the organization present in a binary object, and studying the generation of depth would seem to be a fruitful approach for studying self-organization. Further understanding would also be greatly benefited by the identification of precisely defined computational classes that exhibit self-organization. Biological evolution is a prime example of a self-organizing system. IPCS based systems with selection based on competition among evolving agents is a computational class that encompasses the evolution of DNA. The positive feedback loop for depth generation described previously seems also to be a prime candidate for the mechanism responsible for evolutionary self-organization. If this argument is accepted, then depth provides a natural measure of evolutionary self-organization. By analogy, depth would seem to be strong candidate for characterization of other self-organizing systems. The limitation, is that the definition of depth is restricted to discrete systems.

Discussion

The word evolution is widely used in the popular vernacular to describe almost any situation where change occurs incrementally over time. This broad usage is testimony to the great influence of Darwin's biological theory on modern thought, but confuses understanding of what exactly qualifies as evolutionary -- in the biological sense. A clean definition of biology-like evolution would bring sharpness to many discussions that evoke the idea of evolution. This paper offers such a definition. The definition encompasses biological evolution and evolutionary computer algorithms. There may be other changing systems that also satisfy the definition.

1/21/05

The central claims of the paper are that the evolution of DNA is properly described as a probabilistic iterated computation and that biology-like evolution is defined by the class of iterated probabilistic computations with selection which have selection based on competition among evolving entities. The first claim implies we should look to computational theory for mathematical formalisms to better understand evolution. Logical depth is a measure that was defined to better understand and characterize time constraints on computation. Depth also characterizes the minimal computational effort that is invested in binary objects, and when measured in bits is a complexity measure. Because DNA sequences are informational and can be simply represented in binary, depth also provides measure of DNA sequences. The second claim leads to the conclusion that depth will tend to accumulate in evolutionary systems as a direct consequence of competition among evolving entities. Because of this, it provides a natural measure of DNA sequence complexity.

The foundation of these arguments is a two-part description of life. The first part is the evolution of DNA and second is the formation of organisms. Stadler et al (2001) formalize this division as a trajectory consisting of two maps (figure 1). This scheme isolates the evolution of DNA from the more difficult to characterize processes involving biochemistry, physiology, development, and behavior. Since the information stored in DNA sequences can be simply reduced to binary code and since changes in these sequences are discrete, the evolution of a DNA sequence can be formalized as a computation. Because the change mechanisms are probabilistic, the computation is not deterministic. Because the output of one computation becomes the input for the next, the appropriate class of computations is iterative. Selection is an essential feature of evolution; so a

1/21/05

computational definition of evolution must include selection. Included within the collection of all computations having these features is a large class of activities that are best described as "optimizations." Optimization, though having much in common with evolution, does not capture the essence of biology-like evolution. The class of computations that seems to better capture biology-like evolution consists of those iterated probabilistic computations for which selection is based [at least in part] on competition with other evolving entities. Examples of this class can be created in digital computers and they often exhibit complex and unexpected [to a human observer] behavior.

Biologists naturally tend to see evolution as changes in organisms over time. Identifying the process of DNA change as the essential activity of evolution has the effect of relegating organisms and their associated activities to a secondary position in the scheme of evolutionary change. Organisms become the passive expression of the information contained in the DNA. We are reminded of Richard Dawkins' characterization of the body as nothing more than "the mechanism by which genes reproduce themselves" (Dawkins 1976). It is important to remember, however, that it is organisms, not DNA sequences, that are selected. Thus, the division formalized by the two-part trajectory (figure 1) is not a complete one.

Any newly proposed theory must offer something that old theories do not, and must not reduce the current state of understanding. The paradigm presented in this paper is fully compatible with current evolutionary theory and offers several additional advantages: first, it provides precise guidance about when the term evolution should be used in the biology-like sense by generalizing the notion of biology-like evolution to certain non-biological systems; second, it points to a

1/21/05

computation-based notion of complexity that offers a logical measure of the complexity generated by the evolutionary process; third, it clarifies discussions concerning the generation of diversity in evolutionary systems; and fourth, it simplifies understanding of evolutionary innovation and self-organization.

The scheme has been criticized for relying exclusively on the Turing notion of discrete sequential process. It is certainly true that organisms and their growth, regulation, development and behavior seem better characterized by continuous concurrent process. Whether or not this is simply a perception resulting from the coarse-graining necessitated by our senses of what is actually a discrete universe at the quantum level is a subject beyond this paper. The claims made here avoid the issue by focusing on the evolution of DNA sequences and on non-living systems that are characterized by genomes. This is only possible if the formal partitioning of life into the evolution of DNA and the formation of organisms as illustrated by figure 1 is fundamentally sound. Stadler et al (2001) were led to propose a two-track description of life in order to better understand the evolutionary history of organisms. I was led to similar ideas by the desire to generalize evolutionary theory to non-living systems and to introduce precision in language use.

A criticism from computer scientists may be the reliance on probabilistic computation. In computer science there is no such thing as true randomness. Rylander et al (2001) point out that the Kolmogorov complexity of the output of a genetic program is limited to the Kolmogorov complexity of the various inputs (including the random number generator). The same principle applies to genetic and evolutionary algorithms. For DNA sequences, if the universe is finite and discrete, the limit is the Kolmogorov complexity of the universe. Presumably this is very large.

1/21/05

Quantum mechanics assumes true randomness for quantum phenomena. Thus, by being part of the physical world, DNA sequence change-mechanisms either have access to a source of effective randomness or to true randomness depending on your philosophical bent. In either case the amount of randomness potentially available to evolutionary processes is very large. More fundamental to the arguments presented in this paper is the principle that depth is limited only by the length of computational history. Because of this, the depth of DNA sequences is limited only by the age of life.

Logical depth was defined to formalize a slow-growth restriction that characterizes computation (Bennett, 1988; Jeudes, et al, 1994). When applied to biology the concept of depth seems to naturally characterize organisms and ecosystems. Depth is not a concept that appears in physics, and it is not clear how the depth of physical objects (including organisms) should be defined. If depth does characterize physical objects, then the physical world is also subject to the slow-growth limitation and various features of organism growth, development and evolution would be neatly explained. One simple possibility would allow this. It is that those physical processes relevant to biology (map f in figure 1) are formally describable as computations (the physical Church-Turing hypothesis). If true, then the slow-growth law would also apply to the growth and development of organisms, and deep organisms would only be possible when the inputs to f (DNA sequences and oocyte structure) were deep or the time required for growth and development were long. Clearly, long development times are not favored by natural selection unless something compensates for the extra deaths that occur prior to reproduction. It is tempting to attribute the long development times of complicated organisms to be a consequence of an unrecognized slow-growth law of physical objects. The argument is simple. Complicated

1/21/05

organisms are deep, and prolonged development time is the price that must be paid to produce a deep object from shallower DNA input. The penalty incurred by long development times is more than compensated for by more effective responses to environmental challenges provided by the resultant deep structures. The well-known relationship between brain size and development time is in keeping with this hypothesis. A corollary argument is that there must be constant selection pressure on DNA sequences to become deeper to enable shorter development times. This in turn predicts that DNA sequences should be deep [but not as deep as organisms] after four billion years of selection. Computational theory makes clear that if the f -map is subject to a slow-growth law, then the depth of an organism can be much greater than the depth of its DNA provided sufficient time is allowed for growth and development. Biological theory makes clear that if faster development times are limited by a slow-growth law, then deep DNA sequences will be strongly favored by selection.

It is important to reiterate that there is no accepted concept of depth in current physical theory. So, one must be very careful when speaking about the depth organisms or physical structures except when those structures can be simply and completely represented in digital form.

Finally, an argument has been made by others (Goodwin, 1994) that an additional principle, the generative field, must be included to fully understand biological evolution. It remains unclear whether or not this is so. Addition of a generative field to the definition of biology-like evolution would rule out many evolutionary algorithms, but would not change the basic arguments presented in this paper. The evolutionary trajectory (figure 1) could be drawn with three maps or the f map could be understood to include the generative field. Either way,

1/21/05

computational evolution (map g) remains a fundamental and necessary aspect of biological evolution.

This paper does not attempt to present a finished theory, but rather to lay out a paradigm that makes clear when information theory and computation theory may be justifiably applied to discussions of evolution and when they may not. The paradigm offers a precise definition of biology-like evolution for non-living systems. Considerable space is devoted to the computational concept of depth because the time constraints on the creation of digital objects must also restrict the creation of DNA sequences. This, in turn, represents a fundamental constraint on the evolution of life.

Acknowledgements

I would like to give special thanks to my colleagues Dan Ashlock, Jim Lathrop, and Jack Lutz for many hours of discussion concerning the fascinating subject of complexity.

References

- Ashby, W.B. 1956. Introduction to Cybernetics. John Wiley & Sons, New York.
- Ashlock, D., and J. E. Mayfield. 1998. Acquisition of general adaptive features by evolution. Pages 75-84 in V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben eds. Evolutionary programming VII. Springer, Berlin.
- Aulin, A.Y. 1979. The law of requisite hierarchy. *Kybernetes* 8: 259-266.
- Axelrod, R. 1984. The Evolution of Cooperation. Basic Books, New York.
- Axelrod, R. 1987. The evolution of strategies in iterated prisoner's dilemma. Pages 32-41 in L. Davis ed. Genetic Algorithms and Simulated Annealing. Pitman, South Melbourne.
- Behe, M. J. 1996. Darwin's Black Box. Simon and Schuster, New York.
- Bennett, C. H. 1982. The thermodynamics of computation - a review. *International Journal of Theoretical Physics*, 21:905-940
- Bennett, C. H. 1988. Logical depth and physical complexity. Pages 227-257 in R. Herken, ed. The Universal Turing Machine: a Half-Century Survey. Oxford University Press, Oxford.
- Brooks, D. R., and E. O. Wiley, 1988. Evolution as Entropy: Toward a Unified Theory of Biology", 2nd edition. University of Chicago Press, Chicago.
- Dawkins, R. 1976. The Selfish Gene. Oxford University Press, Oxford.
- Edmonds, B. 1999. What is complexity? – The philosophy of complexity *per se* with application to some examples in evolution. Pages 1-16 in F. Heylighen, J. Bollen and A. Rieger eds. The Evolution of Complexity, the Violet Book of “Einstein meets Magritte.” Kluwer, Dordrecht.
- Fogel, D. B. 1994. An introduction to simulated evolutionary optimization. *IEEE Trans. Neural Networks* 5: 3-14.
- Fogel, D. B. ed. 1998. Evolutionary Computation, The Fossil Record. The Institute of Electrical and Electronics Engineers, New York.
- Goldberg, D. E. 1989. Genetic Algorithms. Addison-Wesley, Reading.
- Goodwin, B. 1994. How the Leopard Changed its Spots. Scribner, New York.

1/21/05

- Gould, S. J. 1988. On replacing the idea of progress with an operational notion of directionality. Pages 319-338 in M. H. Nitecki ed. *Evolutionary Progress*. The University of Chicago Press, Chicago.
- Gould, S. J., and E. S. Vrba. 1982. Exaptation -- a missing term in the science of form. *Paleobiology* 8:4-15.
- Hartl, D. L. 1988. *A Primer of Population Genetics*, 2nd edition. Sinauer Associates, Sunderland.
- Heylighen, F. 1999. The growth of structural and functional complexity during evolution. Pages 17-44 in F. Heylighen, J. Bollen and A. Rieger eds. *The Evolution of Complexity, the Violet Book of "Einstein meets Magritte."* Kluwer, Dordrecht.
- Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor.
- Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems*, 2nd edition. MIT Press, Cambridge.
- Holland, J. H. 1998. *Emergence*. Addison-Wesley, Reading.
- Jaynes, E. T. 1978. Where do we stand on maximum entropy? Pages 15-118. in R. D. Levine and M. Tribus, eds. *The Maximum Entropy Formalism*. MIT Press, Cambridge.
- Juedes, D. W., J. I. Lathrop, and J. H. Lutz. 1994. Computational depth and reducibility. *Theoretical Computer Science* 132:37-70.
- Kauffman, S. A. 1993. *The Origins of Order*. Oxford University Press, Oxford.
- Kauffman, S. A. 1995. *At Home in the Universe*. Oxford University Press, Oxford.
- Lathrop, J. I. 1997. *Computing and Evolving Variants of Computational Depth*. Ph.D. Dissertation, Iowa State University, Ames.
- Lathrop, J. I. 1998. Compression depth and genetic programs. In John Koza, ed., *Genetic Programming 1997*, pp 370-378. Morgan Kaufmann, San Francisco.
- Lempel, A. and J. Ziv. 1978. Compression of individual sequences via variable rate coding. *IEEE Transactions on Information Theory* 24:530-536.
- Lewin, R. 1992. *Complexity: Life at the Edge of Chaos*. Macmillan, New York.

1/21/05

- Li, M. and P. Vitanyi, 1997. *An Introduction to Kolmogorov Complexity and Its Applications*. 2nd ed. Springer Verlag, Berlin.
- McShea, D. W. 1994. Complexity and evolution: what everybody knows. *Biology and Philosophy* 6:303-324.
- McShea, D. W. 1996. Metazoan complexity and evolution: is there a trend? *Evolution* 50:477-492.
- Miettinen, K., P. Neittaanmaki, M. M. Makela, and J. Periaux, eds. 1999. *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming, and Industrial Applications*. Wiley, New York.
- Miller, J.H. 1996. The coevolution of automata in the repeated prisoner's dilemma. *J. Economics Behavior and Organization* 29:87-112.
- Nitecki, M.H. 1990. *Evolutionary Innovation*. U. Chicago Press. Chicago.
- Raup, D.M. 1994. The role of extinction in evolution. *Proc. Nat Acad. Sci. (USA)* 91: 6758-6763.
- Rylander, B., T., Soule, and J. Foster. 2001. Computational complexity, genetic programming, and implications. Pages 348-360 *in* Julian Miller, ed., *Genetic Programming: 4th European Conference, EuroGP2001*. Springer, Berlin.
- Shannon, C. E., and W. Weaver. 1949. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana.
- Smolin, L. 1997. *The Life of the Cosmos*. Oxford University Press, Oxford.
- Stadler, B.M.R., P. F. Stadler, and W. Fontana, 2001. The topology of the possible: formal spaces underlying patterns of evolutionary change. *J. Theoretical Biology* 213: 241-274.
- Stanley, E.A., D. Ashlock, and L. Tesfatsion. 1994. Iterated prisoner's dilemma with choice and refusal. Pages 131-137 *in* Christopher Langton, ed., *Artificial Life III*, v. 17 of Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley, Reading.
- Van Valen, L. 1973. A new evolutionary law. *Evolutionary Theory* 1: 1-30.
- Wolfram, S. 1983. Statistical mechanics of cellular automata. *Reviews of Modern Physics* 55:601-644.

1/21/05

Wolfram, S. 2002. A New Kind of Science. Wolfram Media, Champaign.