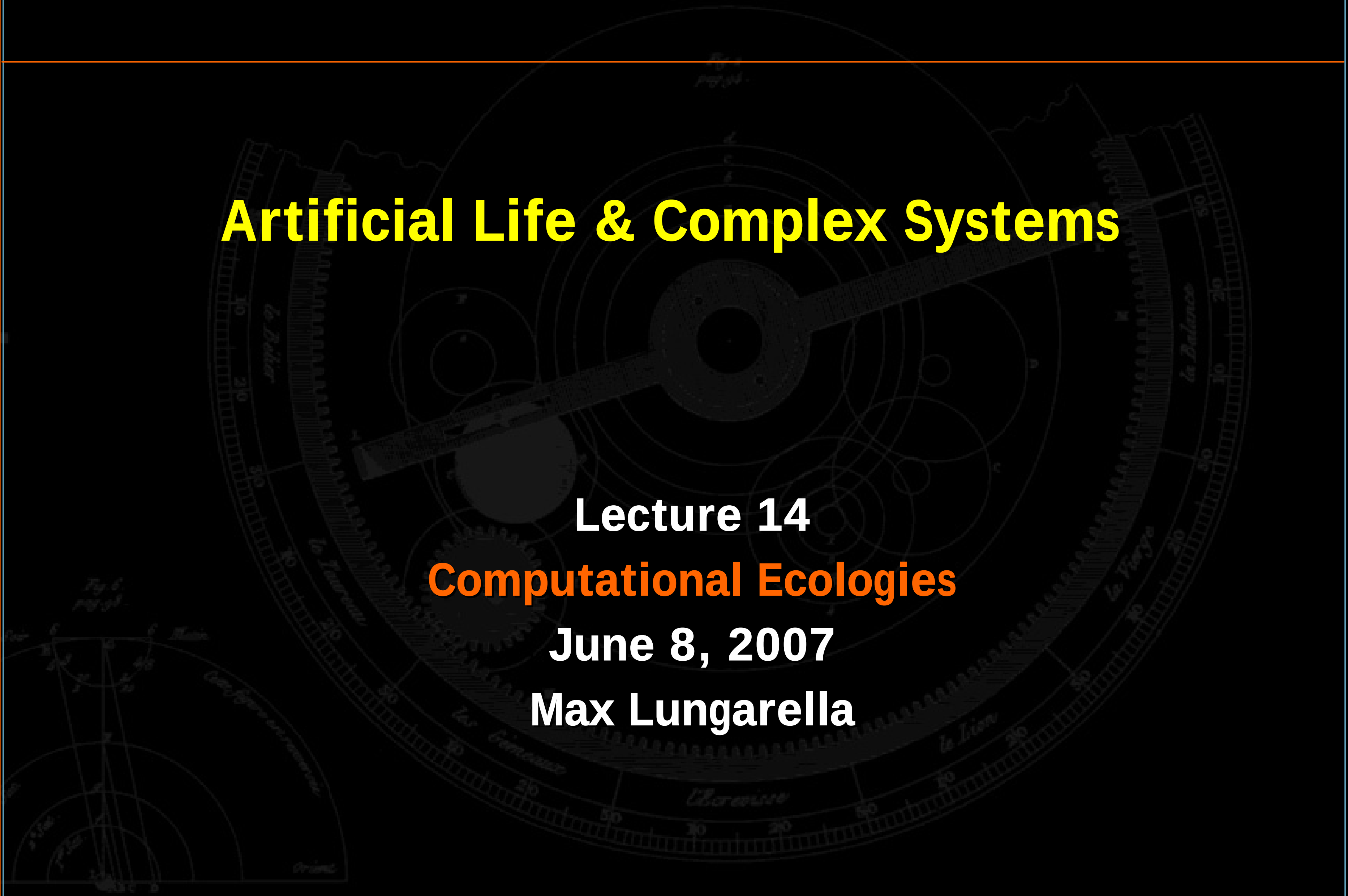


Artificial Life & Complex Systems

Lecture 14

Computational Ecologies

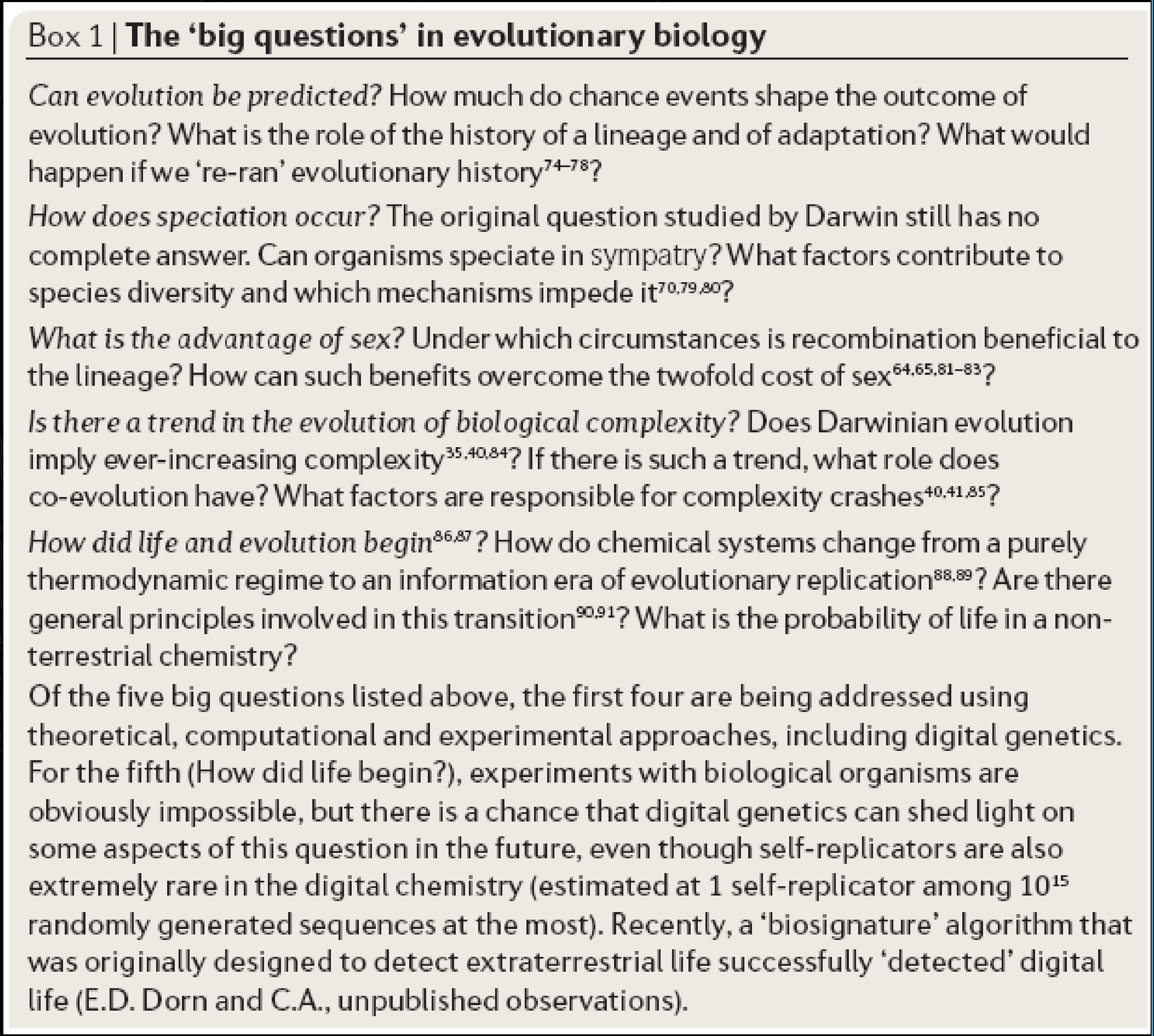
June 8, 2007

Max Lungarella

Contents

- Digital organisms
- Core Wars
- Tierra
- Avida
- PolyWorld
- SimLife

Big Questions



Box 1 | **The 'big questions' in evolutionary biology**

Can evolution be predicted? How much do chance events shape the outcome of evolution? What is the role of the history of a lineage and of adaptation? What would happen if we 're-ran' evolutionary history[74–78]?

How does speciation occur? The original question studied by Darwin still has no complete answer. Can organisms speciate in sympatry? What factors contribute to species diversity and which mechanisms impede it[70,79,80]?

What is the advantage of sex? Under which circumstances is recombination beneficial to the lineage? How can such benefits overcome the twofold cost of sex[64,65,81–83]?

Is there a trend in the evolution of biological complexity? Does Darwinian evolution imply ever-increasing complexity[35,40,84]? If there is such a trend, what role does co-evolution have? What factors are responsible for complexity crashes[40,41,85]?

How did life and evolution begin[86,87]? How do chemical systems change from a purely thermodynamic regime to an information era of evolutionary replication[88,89]? Are there general principles involved in this transition[90,91]? What is the probability of life in a non-terrestrial chemistry?

Of the five big questions listed above, the first four are being addressed using theoretical, computational and experimental approaches, including digital genetics. For the fifth (How did life begin?), experiments with biological organisms are obviously impossible, but there is a chance that digital genetics can shed light on some aspects of this question in the future, even though self-replicators are also extremely rare in the digital chemistry (estimated at 1 self-replicator among 10^{15} randomly generated sequences at the most). Recently, a 'biosignature' algorithm that was originally designed to detect extraterrestrial life successfully 'detected' digital life (E.D. Dorn and C.A., unpublished observations).

Adami, C. (2006)

The (Evolutionary) Dynamics of Adaptation

- We are still far from a satisfactory understanding of the genetic bases of evolutionary change and adaptation
- Experiment has not kept track of theoretical developments because <u>quantitative</u> experiments are difficult

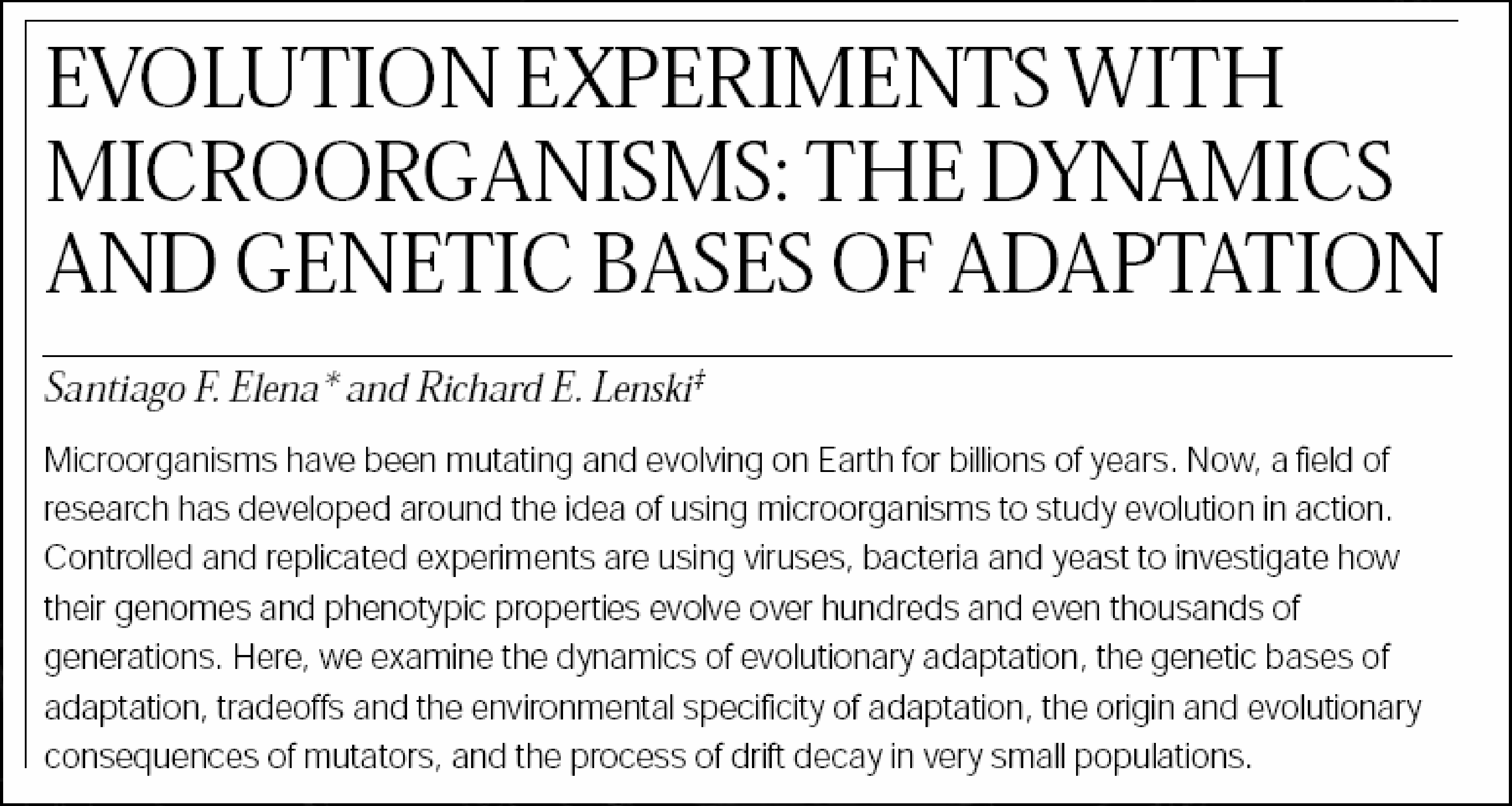
Evolution and Universality

- Theory of evolution's claim to <u>universality</u>: Results of experiments are independent of "vector" (bacterium, fungus, plant, or animal) used
- Not all organisms are suitable for controlled, quantitative experiments

Requirement for Experimental Organisms

- Abundant, easy to breed
- Short generational time
- Simple environment that can be controlled and manipulated
- Ease of measurement, e.g. fitness, genomics
- Ease of storage of historical lineages

Evolution Experiments with μ-Organisms



EVOLUTION EXPERIMENTS WITH MICROORGANISMS: THE DYNAMICS AND GENETIC BASES OF ADAPTATION

Santiago F. Elena and Richard E. Lenski‡*

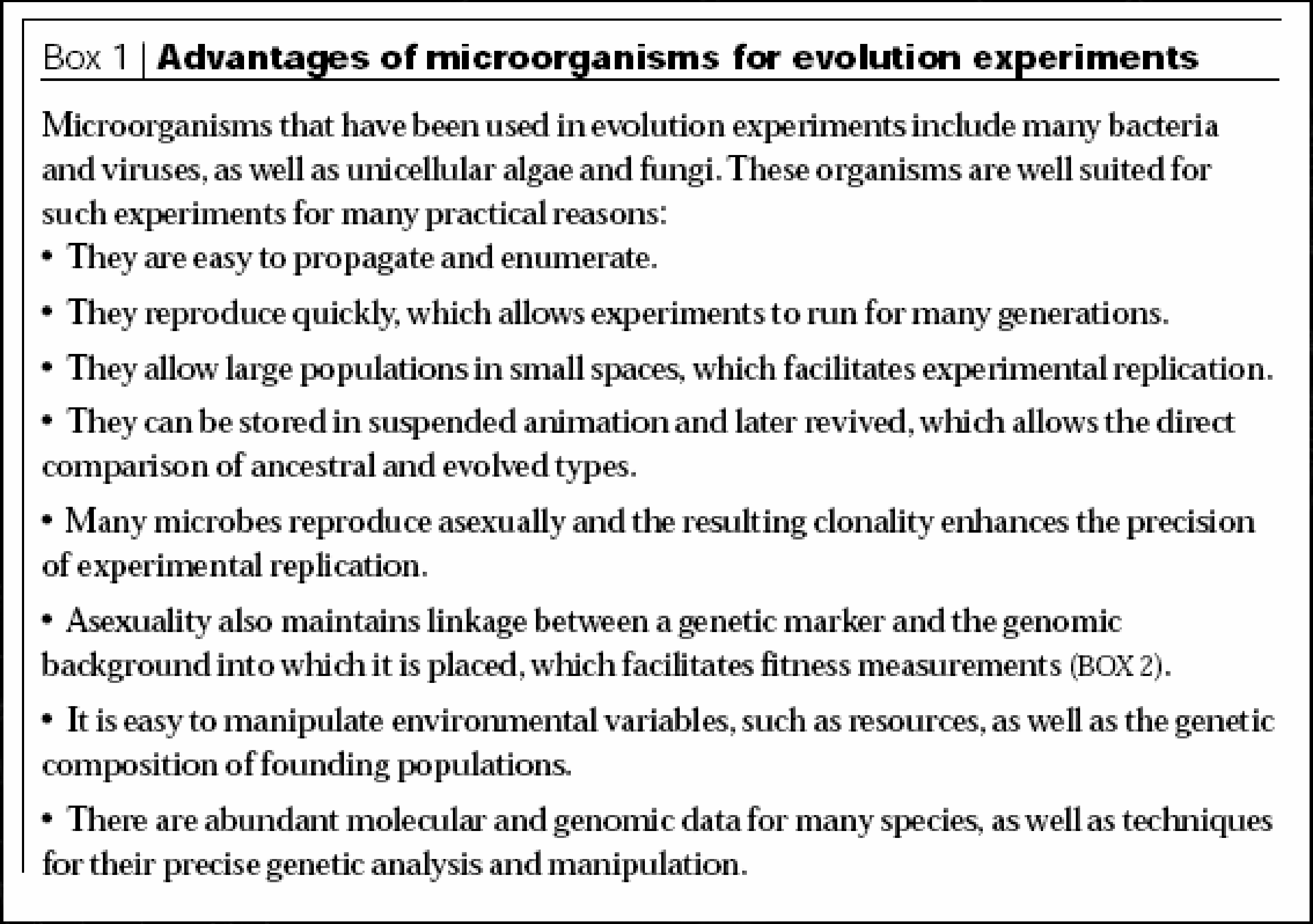
Microorganisms have been mutating and evolving on Earth for billions of years. Now, a field of research has developed around the idea of using microorganisms to study evolution in action. Controlled and replicated experiments are using viruses, bacteria and yeast to investigate how their genomes and phenotypic properties evolve over hundreds and even thousands of generations. Here, we examine the dynamics of evolutionary adaptation, the genetic bases of adaptation, tradeoffs and the environmental specificity of adaptation, the origin and evolutionary consequences of mutators, and the process of drift decay in very small populations.

Elena, S.F. and Lenski, R.E. (2003) Nature Reviews Genetics

Experimental Organisms

- Viruses (Φ6, ΦX174,VSV)
- Bacteria (E. coli, Myxococcus, Pseudomonas, Salmonella)
- Fungi (Yeast)
- Green algae (Chlamydomonas)
- Flatworms (C. elegans)
- Insects (Drosophila)
- Higher organisms (Guppies, zebrafish,...)

Evolution Experiments with μ-Organisms



Box 1 | **Advantages of microorganisms for evolution experiments**

Microorganisms that have been used in evolution experiments include many bacteria and viruses, as well as unicellular algae and fungi. These organisms are well suited for such experiments for many practical reasons:

- They are easy to propagate and enumerate.
- They reproduce quickly, which allows experiments to run for many generations.
- They allow large populations in small spaces, which facilitates experimental replication.
- They can be stored in suspended animation and later revived, which allows the direct comparison of ancestral and evolved types.
- Many microbes reproduce asexually and the resulting clonality enhances the precision of experimental replication.
- Asexuality also maintains linkage between a genetic marker and the genomic background into which it is placed, which facilitates fitness measurements (BOX 2).
- It is easy to manipulate environmental variables, such as resources, as well as the genetic composition of founding populations.
- There are abundant molecular and genomic data for many species, as well as techniques for their precise genetic analysis and manipulation.

Elena, S.F. and Lenski, R.E. (2003) Nature Reviews Genetics

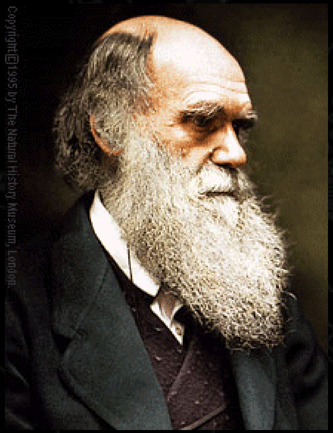
E. Coli Long-Term Experiment

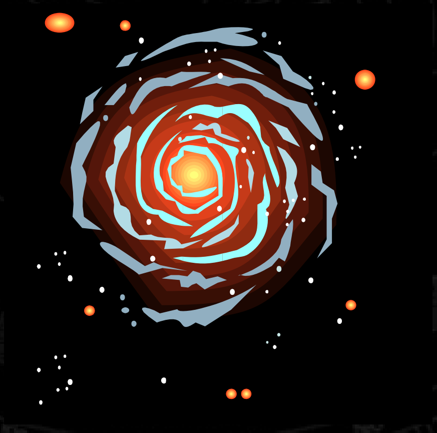
Richard Lenski (Michigan State U.) started evolution experiment with twelve identical lines of *E. coli* bacteria in 1990 to study adaptation, divergence, macroevolution

After 20,000 generations of evolution, experiment is still ongoing

Evolution and Universality

- Is dynamics of evolutionary process truly independent of organism?
- Does Darwinian evolution occur for every form of life anywhere in the universe?





Evolution and Universality

Hypothesis can be tested if:

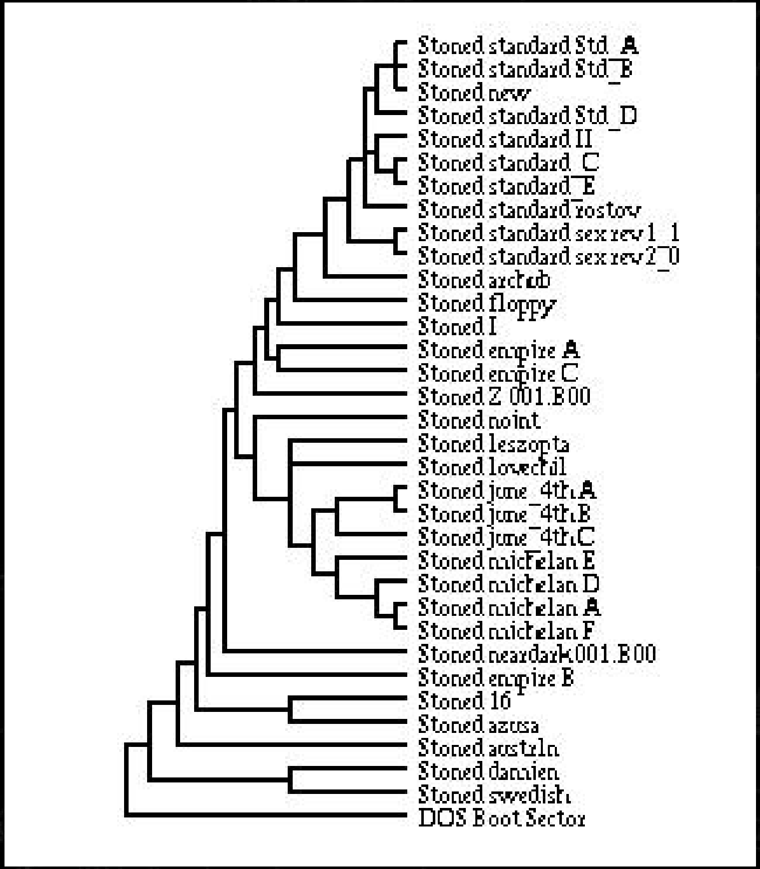
- Discover new form of life in solar system, meteorite, or star
- Create an artificial form of life

Only second option is viable:

"So far, we have been able to study only one evolving system and we cannot wait for interstellar flight to provide us with a second. If we want to discover generalizations about evolving systems, we have to look at artificial ones" *(John Maynard Smith, 1992)*

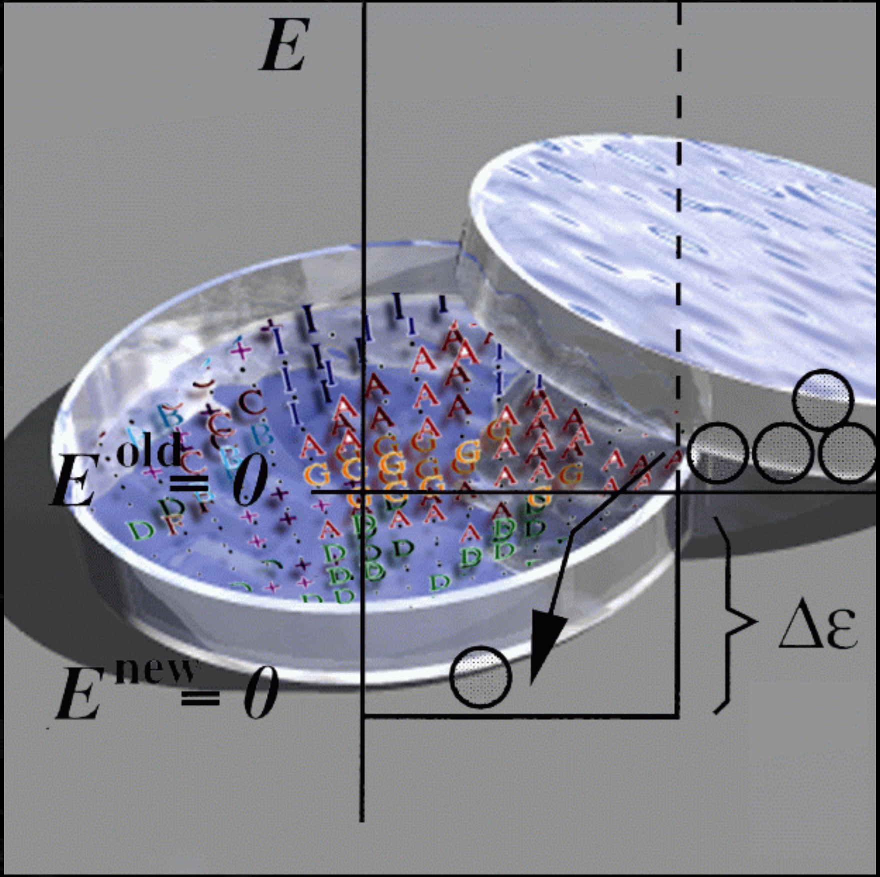
Digital Life

Computer viruses ("Stoned" family)



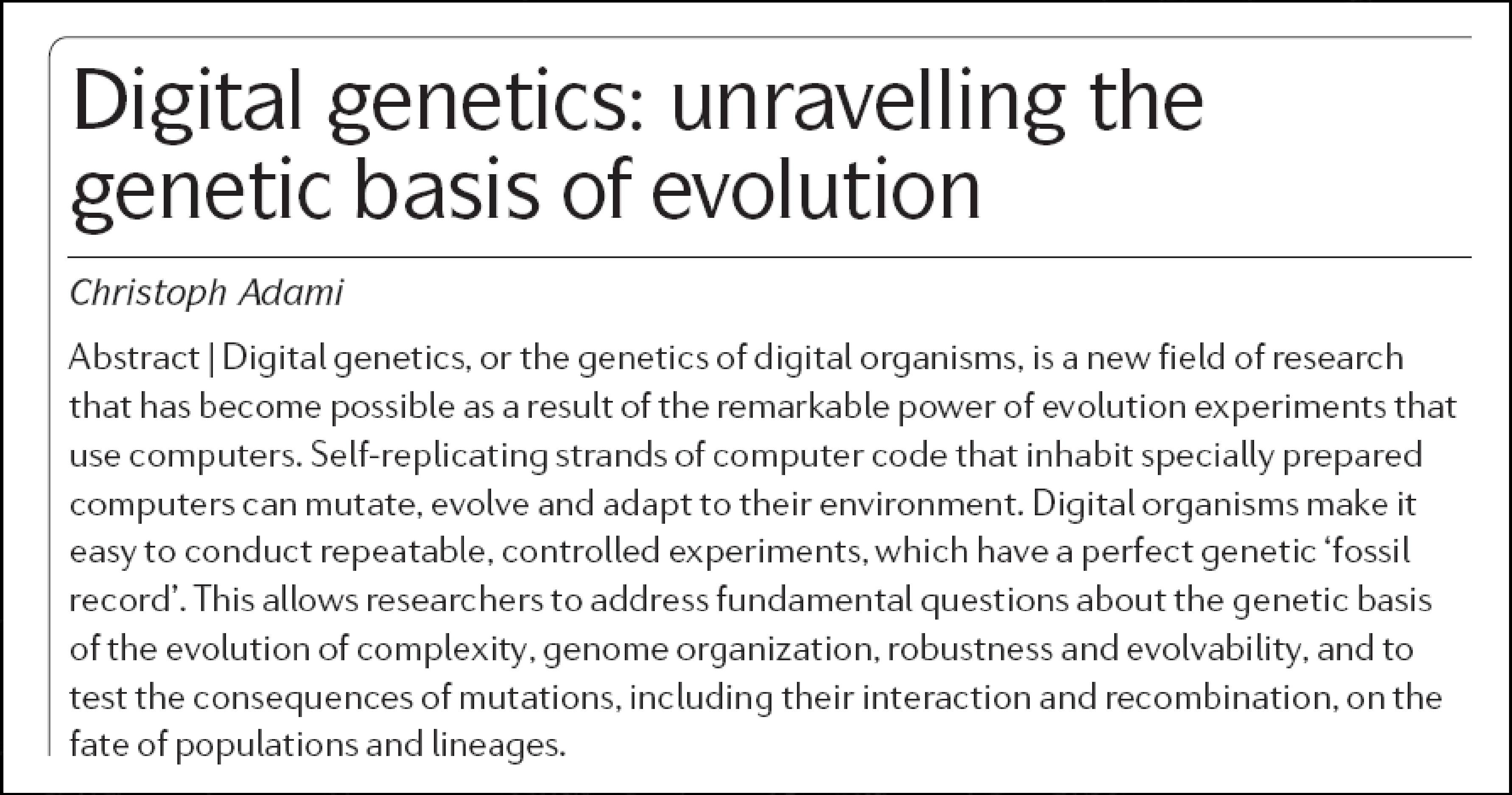
D.H.Hull

Domesticated, adapting "viruses"



Adami, C. (1997)

Computational Ecologies (or Digital Life)



Digital genetics: unravelling the genetic basis of evolution

Christoph Adami

Abstract | Digital genetics, or the genetics of digital organisms, is a new field of research that has become possible as a result of the remarkable power of evolution experiments that use computers. Self-replicating strands of computer code that inhabit specially prepared computers can mutate, evolve and adapt to their environment. Digital organisms make it easy to conduct repeatable, controlled experiments, which have a perfect genetic 'fossil record'. This allows researchers to address fundamental questions about the genetic basis of the evolution of complexity, genome organization, robustness and evolvability, and to test the consequences of mutations, including their interaction and recombination, on the fate of populations and lineages.

Adami, C. (2006) Nature Reviews Genetics

Where It All Started: Core Wars

- Dewdney (1984) “Computer recreations” Scientific American (inspired by a game called “Darwin” written by Vyssotsky, Morris and Ritchie in the 1960s)
- Programming game where assembly programs try to destroy each other in the memory of a simulated computer
- The programs (warriors) are written in a special low-level language called Redcode, and run by a program called MARS (Memory Array Redcode Simulator)
- The warriors fight for control of resources (memory and processor cycles)
- Goal: cause all opposing programs to terminate leaving the winner in sole possession of machine

http://www.corewars.org

Core Wars

- The Core (the memory of the simulated computer) is a continuous array of instructions, empty except for the competing programs
- The Core (the virtual ring) wraps around
- Each battle program occupies one location in Core
- MARS runs battle program by alternatively executing one instruction of each (time-sharing system)

Redcode – Instruction Set

DAT	data (kills the process)
MOV	move (copies data from one address to another)
ADD	add (adds one number to another)
SUB	subtract (subtracts one number from another)
MUL	multiply (multiplies one number with another)
DIV	divide (divides one number with another)
MOD	modulus (divides one number with another and gives the remainder)
JMP	jump (continues execution from another address)
JMZ	jump if zero (tests a number and jumps to an address if it's 0)
JMN	jump if not zero (tests a number and jumps if it isn't 0)
DJN	decrement and jump if not zero (decrements a number by one, and jumps unless the result is 0)
SPL	split (starts a second process at another address)
CMP	compare (same as SEQ)
SEQ	skip if equal (compares two instructions, and skips the next instruction if they are equal)
SNE	skip if not equal (compares two instructions, and skips the next instruction if they aren't equal)
SLT	skip if lower than (compares two values, and skips the next instruction if the *first* is lower than the *second*)
LDP	load from p-space (loads a number from private storage space)
STP	save to p-space (saves a number to private storage space)
NOP	no operation (does nothing)

Typical Strategies

1) Imp: executed moves its way through the array at a speed of one address per cycle leaving behind a trail of instructions
2) Juggernaut: copies itself 10 locations ahead
3) Dwarf: works its way through the memory array bombarding every 5th address with zero (zero = non-executable data)

Core Wars

Life Demo!

The Coreworld System

Rasmussen et al., 1990

Coreworld = simulated computer core arranged in a circular manner

Space is seeded with assembly-like instructions of the original Core War language (Redcode)

Core is subject to noise (MOV command is flawed with a certain probability)

Goal: construct an artificial chemistry that could bear replicating artificial molecules, in the hope that the molecules would display emergent computational behavior

Tierra – Studying Evolution in Silico

Biologist by training

- Specialized in plant ecology (tropical ecologist)
- Studied vines, ants, butterflies, and beetles
- Rainforests in Costa Rica

Began modeling evolutionary processes in 1990

- Desire to observe evolutionary process in a medium other than carbon chemistry
- Current knowledge of life and evolution is based on a sample size of one
- Evolves computer programs (digital organisms)
- CPU time plays role of “energy” resource
- Computer memory is the “material” resource
- Study structure, behavior of genetic lines over time
- Evolved parasitism, arms races, symbiosis, etc.

Ray (1990)

Tierra

- Software that creates environment inside of a computer for populations of *self-replicating* programs, subject to *mutation* and *survival of the fittest* (pure Darwinism)
- Tierra ‘runs’ are *experiments*, not simulations. The programs exist, adapt to their world, and *grow* in complexity
- Experiments can be *repeated* many times for statistical accuracy
- Presence of active co-evolutionary dynamics; there is no explicitly defined fitness function; fitness emerges as a result of the relative ability of genetic variants to survive and reproduce

RNA World

"Central Dogma" of modern biology:

- DNA → RNA → protein

However: "chicken and egg" problem:

- DNA requires protein enzymes to copy itself
- Proteins are synthesized by DNA

RNA comes to the rescue

- RNA can act as a catalytic enzyme (to produce copies of itself)
- Hence the theory of an early "RNA World"

Instructions in an RNA World

Tierra's <u>machine instructions</u> are analogous to amino acids in an RNA world

- They are the fundamental code
- They are <u>chemically active</u>

Tierra's <u>digital organisms</u> are analogous to RNA (Ray calls them "creatures") in an RNA world

- They embody the "genetic" code
- They <u>carry out all metabolic activity</u>

Initialization and Labeling

- A block of computer RAM is designated as a "soup"
- The soup is inoculated with a seed creature of 80 instructions, that is able to do nothing but copy itself
- The creature's genotype is these 80 instructions
- Evolution is tracked by following genetic lineages
  - Genotypes are labeled as

    <# of instructions> <distinct instruction set identifier>

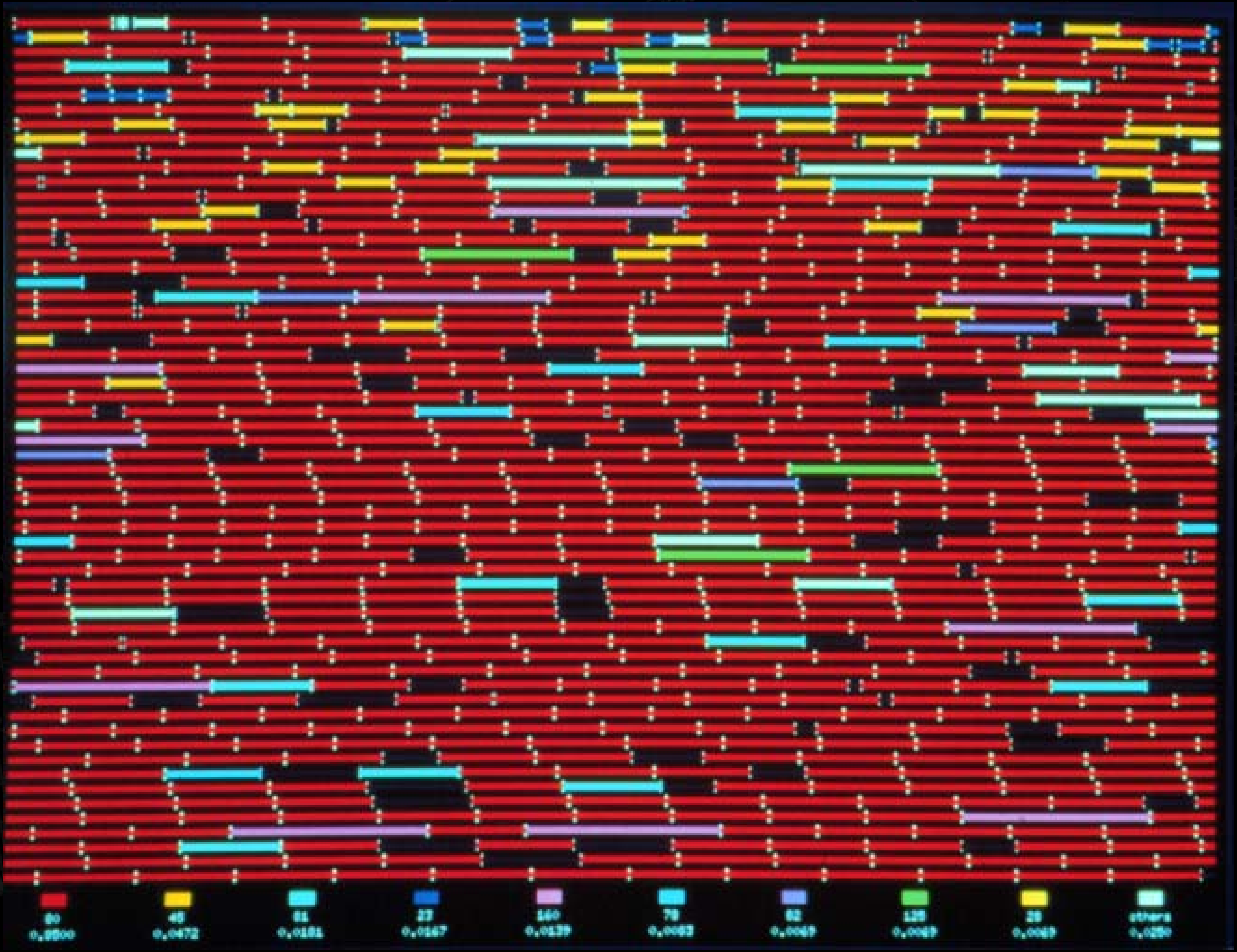
    80aaaa

    80aaab

    45aaaa

    ...

The Soup


80
0.8500
45
0.0472
81
0.0181
27
0.0167
160
0.0139
79
0.0083
82
0.0069
125
0.0069
28
0.0069
others
0.0250

Tierran Language

Size of instruction set is comparable to size of genetic code

- DNA uses 64 codons that are translated into 20 amino acids
- Tierran language uses 32 instructions, *including operands*
  - Accomplished by addressing only registers and stack

"Address by template" (branch to pattern)

- Real molecular interactions are based on patterns/templates, not locations
- Templates are built by inline NOP_0 and NOP_1 instructions

(e.g. *jmp np0 np1 np1* will jump to next instance of *np1 np0 np0*)

Errors are ignored (except for effect on "reaping")

Tierran Creatures

Size is just the size of its allocated block of memory (usually the number of instructions)

Cells have "semi-permeable membranes"

- Only the creature itself has write permission
- Any creature has read permission in any other

During reproduction, parent cell has write permission to newly allocated memory of daughter cell

- Write permission is surrendered when daughter cell is made independent of parent

Creatures Have a Virtual CPU

Contains

- 2 address registers + 2 numeric registers
- Small Stack + Stack pointer
- Instruction pointer
- Flags register to indicate error conditions

Performs fetch-decode-execute-inc(IP) cycle

Has a simple instruction set for

- Arithmetics, bit manipulation
- Moving data between registers and RAM
- Control "instruction pointer" (IP)

Computations are probabilistic

- **Mutations** occur at some low rate

Time Resource Management

- Virtual CPUs are maintained in a circular queue to be executed in sequence on the real CPU
- Number of Tierran instructions executed in each time slice can be made proportional to creature size, inversely proportional to creature size, or invariant with respect to create size
    - Proportionality constant controls whether selection favors large or small creatures, or is size neutral
    - In practice, since the ability to reproduce in fewer cycles conveys a reproductive advantage on creatures, Tierra tends to evolve smaller and smaller programs

The Reaper

Since memory rapidly fills up, a reaper kills less fit creatures

All creatures are on the reaper's linear queue

- At birth creatures enter the bottom of the queue
- The reaper kills creatures at the top of the queue
- Creatures move up in the queue when they generate an error condition (unless the creature above them has more errors)
- Creatures move down in the queue when they execute certain difficult instructions (e.g. div or mul) successfully (unless the creature below them has fewer errors)

Enter Variation

Mutation (single bit)

- At a low rate (1 bit/10,000 instructions executed) a random bit from anywhere in the soup is flipped
- At a higher rate (1 bit/1,000 to 2,500 instructions copied) a bit is flipped during a copy instruction (normally associated with a creature reproducing)

Probabilistic execution (random replacements)

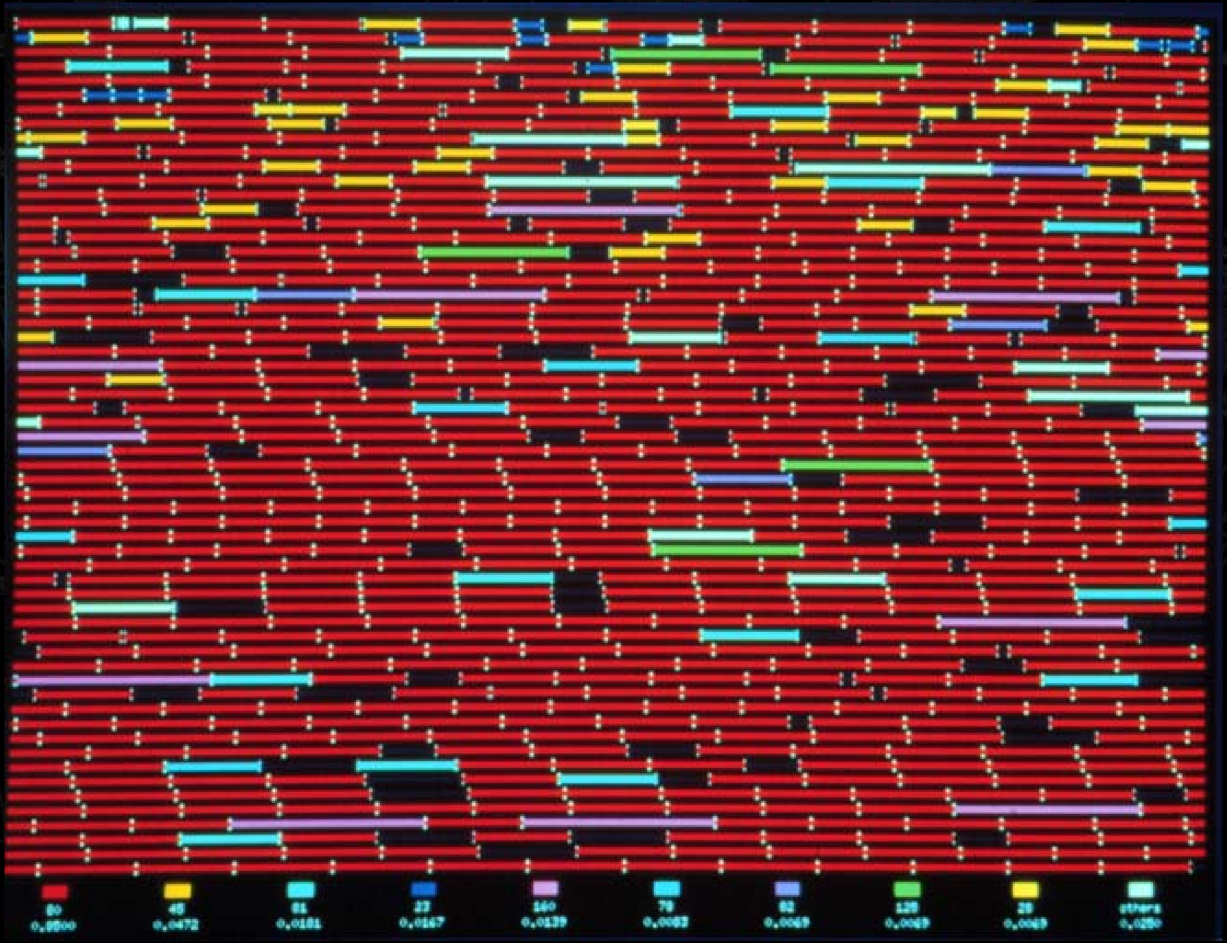
- At a low rate (not specified) Tierran instructions are executed imperfectly (random instruction)
  - Usually the same instruction is executed, but off by one
    - Increment by one actually increments by zero or two
    - Bit flipping instruction flips second bit instead of first
    - Shift instructions shifts two bits instead of one

Evolution in Tierra

- Parasites evolve that have no copy loop, instead using the copy loop of a host organism; their smaller size confers a speed advantage
- Immunity to parasites evolves in the host population
- New parasites emerge that are able to circumvent this immunity (hyper-parasites parasitize the parasites, forcing the parasites to copy the hyper-parasites)
- Social hyper-parasites evolve which can only reproduce in communities of hyper-parasites
- Cheaters (hyper-hyper-parasites) then subvert the social convention
- Time course is one of "punctuated equilibrium"

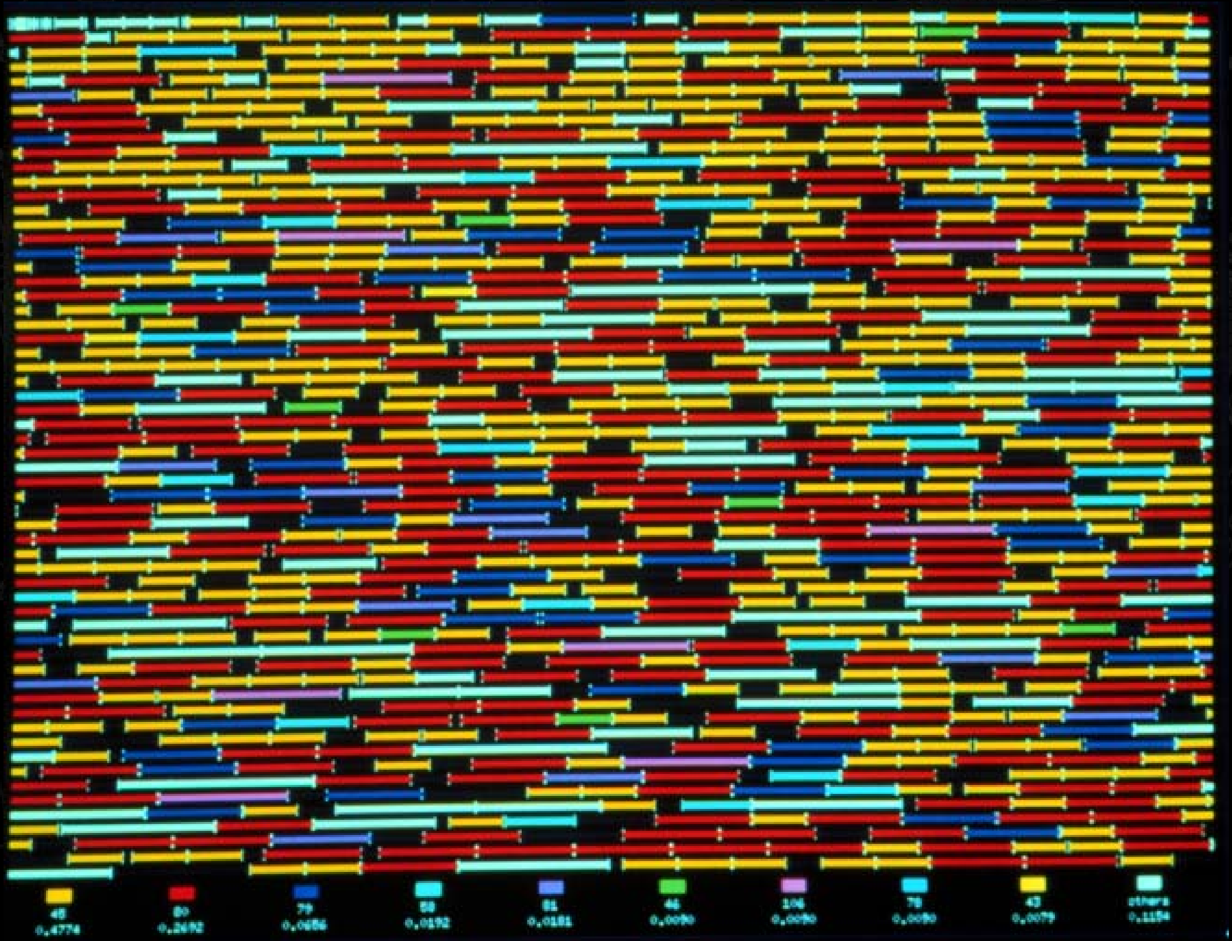
Evolution in Tierra

Creatures represented by colored bars; colors indicate genome size (e.g., red = 80 bytes, yellow = 45 bytes, blue = 79 bytes)



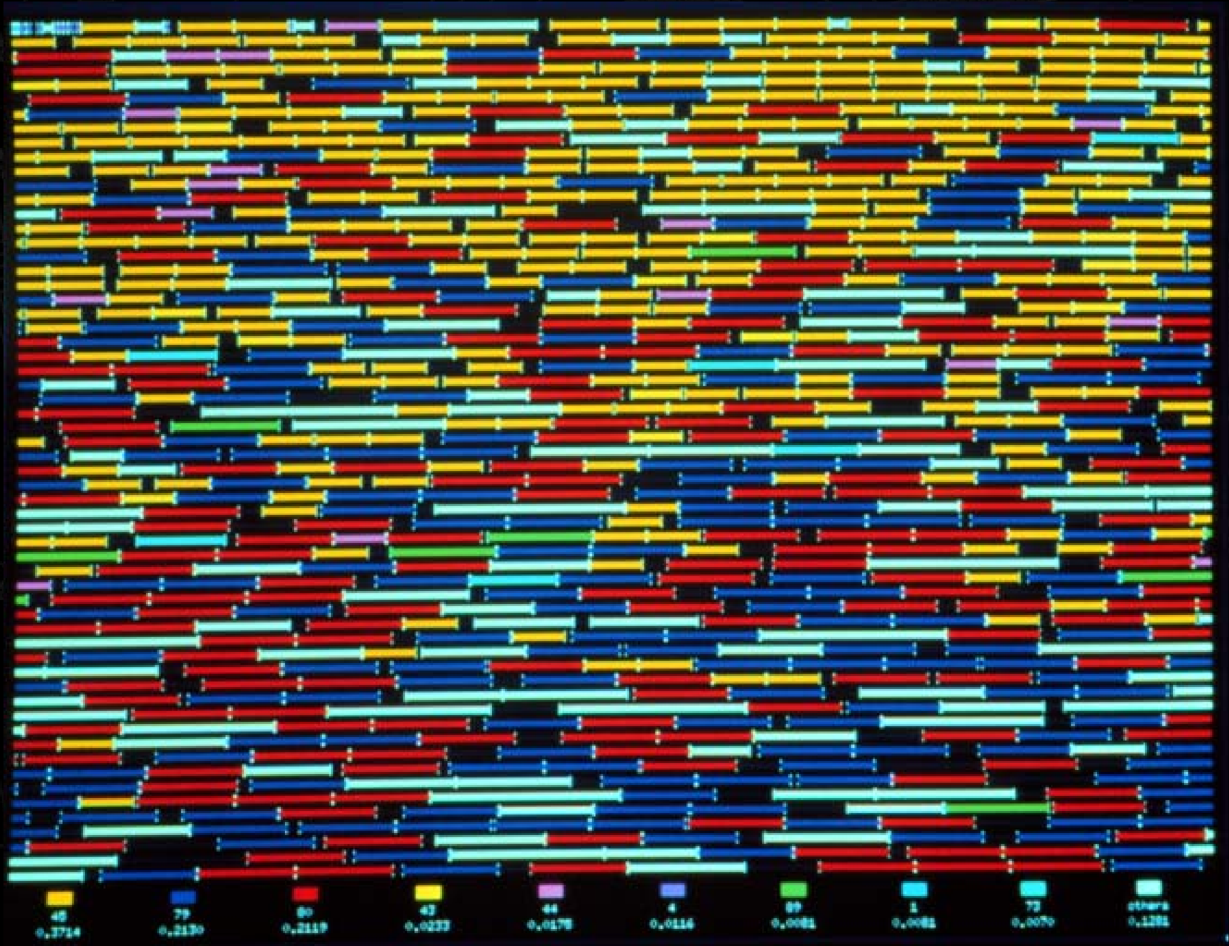
Evolution in Tierra

Creatures represented by colored bars; colors indicate genome size (e.g., red = 80 bytes, yellow = 45 bytes, blue = 79 bytes)



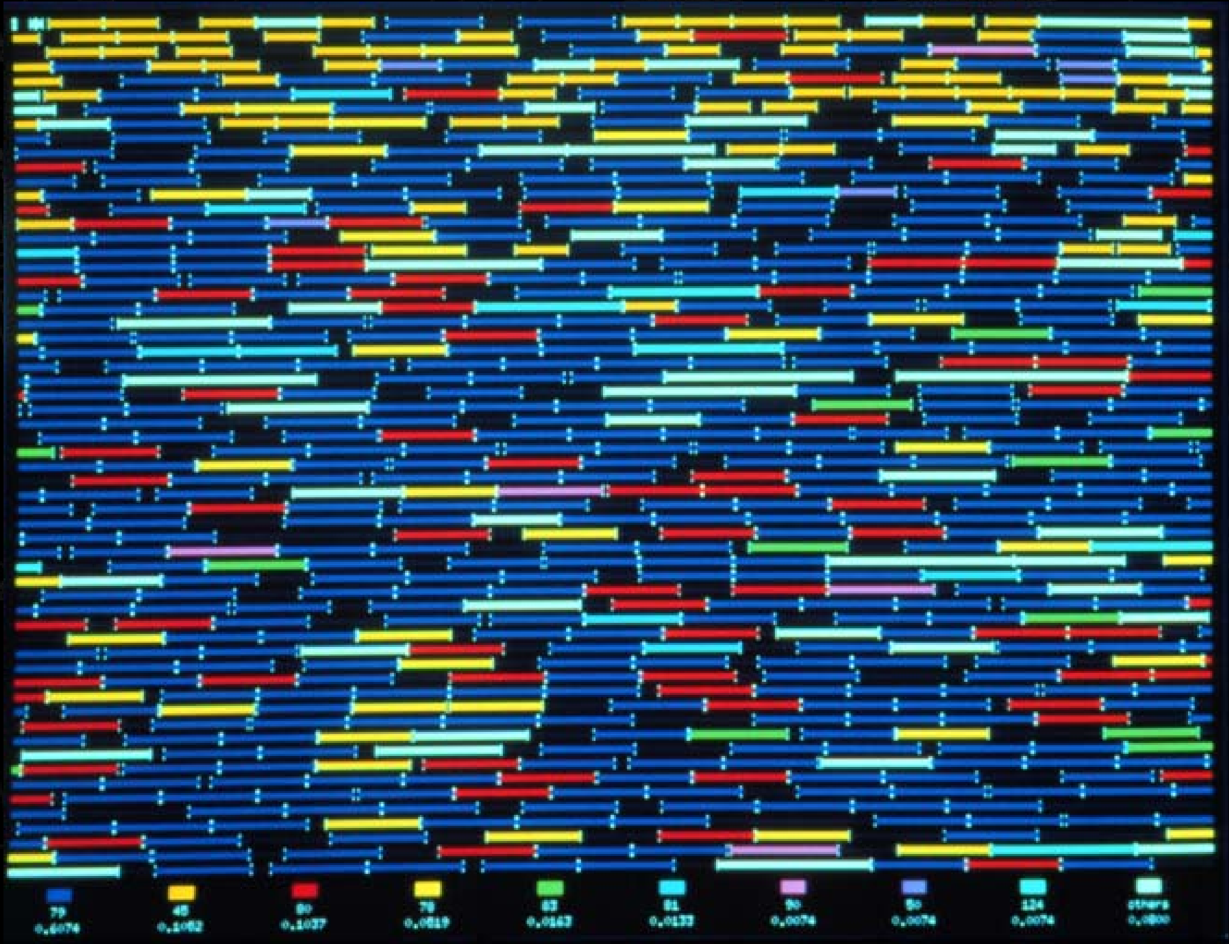
Evolution in Tierra

Creatures represented by colored bars; colors indicate genome size (e.g., red = 80 bytes, yellow = 45 bytes, blue = 79 bytes)



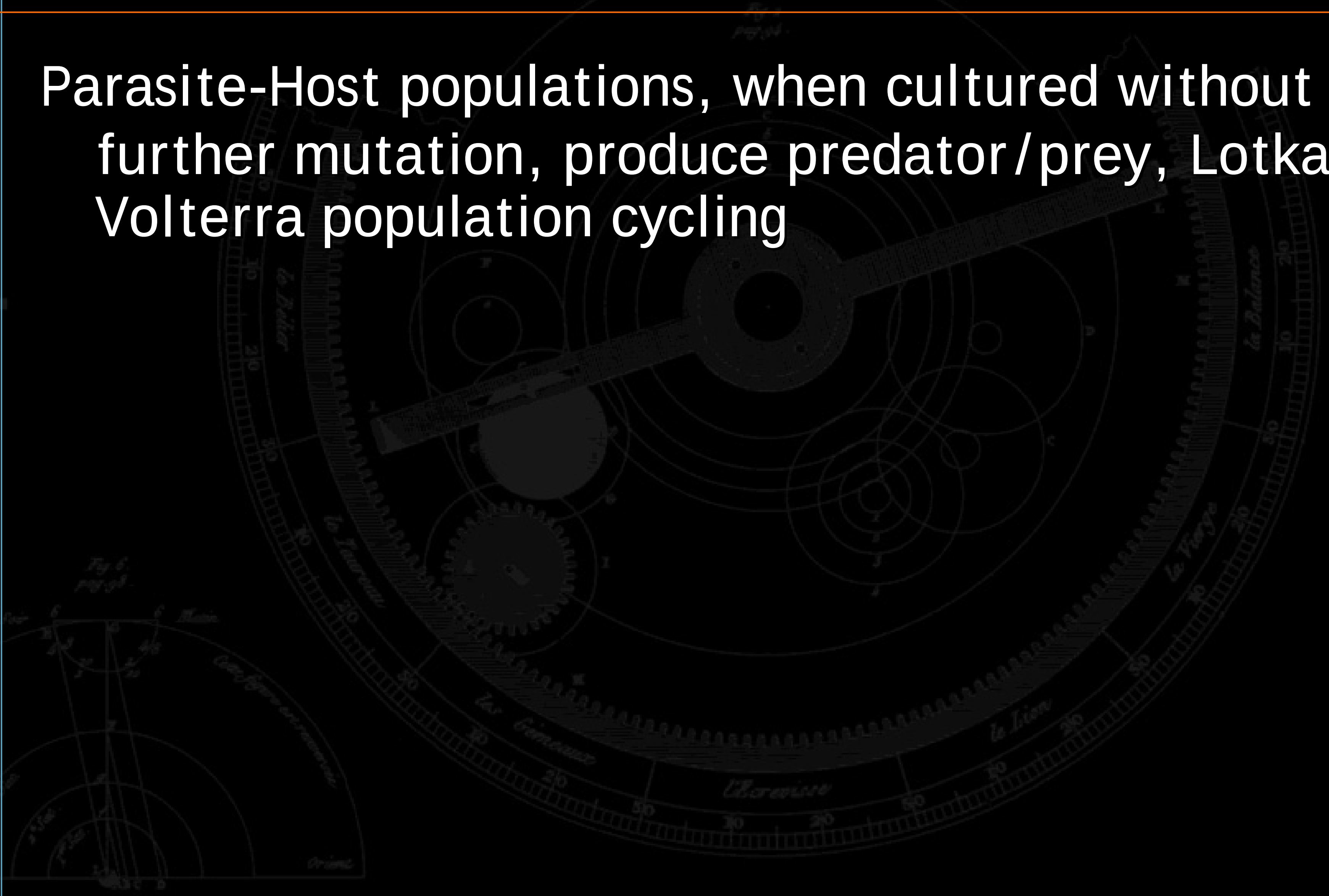
Evolution in Tierra

Creatures represented by colored bars; colors indicate genome size (e.g., red = 80 bytes, yellow = 45 bytes, blue = 79 bytes)



Evolution in Tierra

Parasite-Host populations, when cultured without further mutation, produce predator/prey, Lotka-Volterra population cycling



Tierra

Movie Time: Let Tom Ray explain Tierra

Avida: Avidans

- Each gene is made up sets of instructions that carry a particular function
- Language has between 20 and 30 instructions
- Language is quite robust to mutations (20%-70% of single-instruction substitutions do not affect program’s function)

Related Computational Ecologies

Network Tierra (T. Ray)

- Connect many machines together to form a bigger “soup”

Avida (Adami, Brown, 1994), similar idea but

- On a grid (locality)
- I/O and (limited) ability to train organisms to perform functions
- Active research

Amoeba (Pargellis, 1996), similar idea

- Simpler instruction set
- Spontaneous emergence of self-replicators

Physis (A. Egri-Nagy, 2003)

- Evolves both: VM and programs
- Encodes the computer together with the program

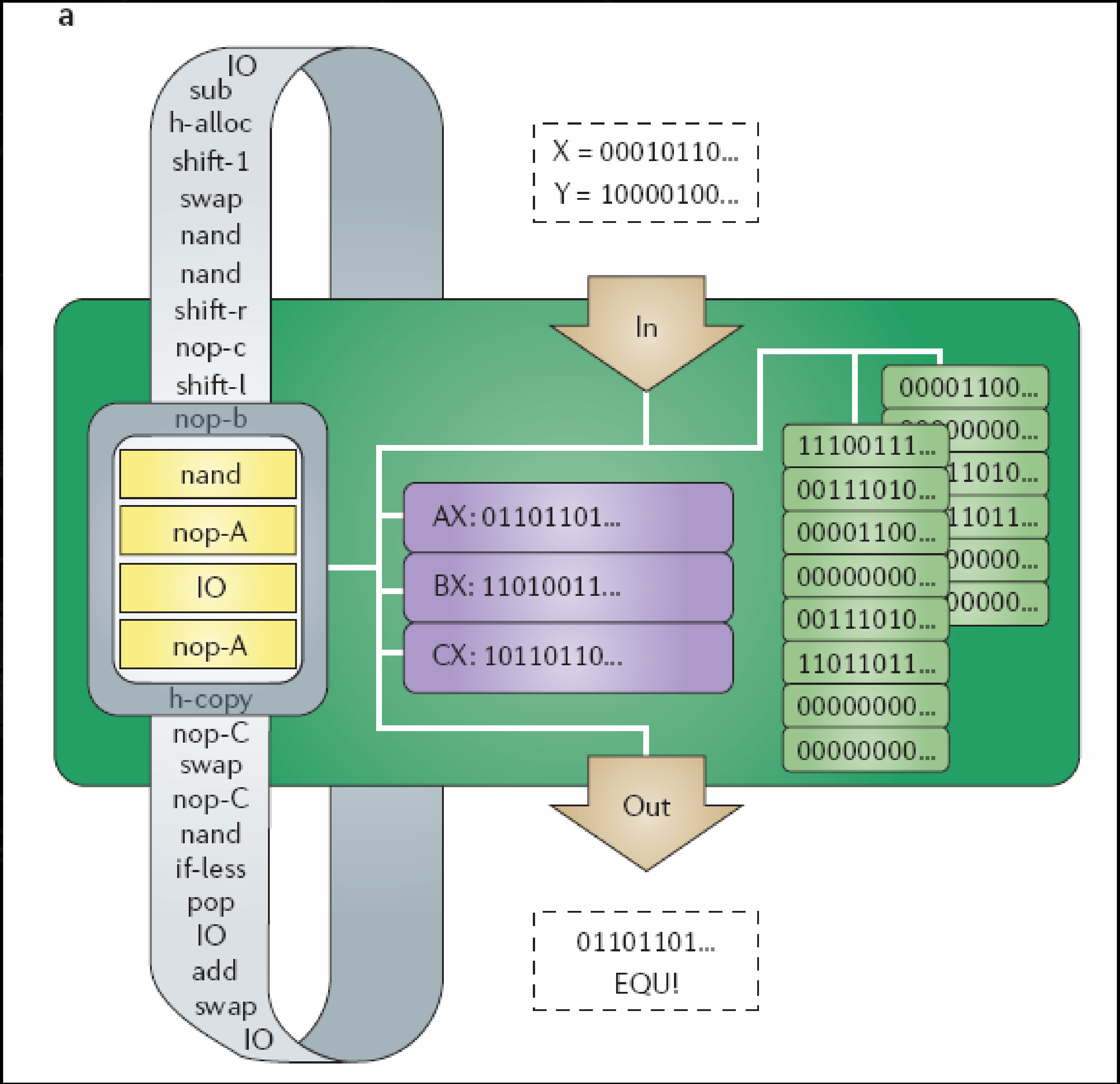
String-Based Tierra (K. Sigiura, 2003)

- Encodes programs into strings
- Uses reg-expr rules to match-and-substitute (to compute)
- Rules are strings as well
- Evolve programs and their rules as a single individual

Avida

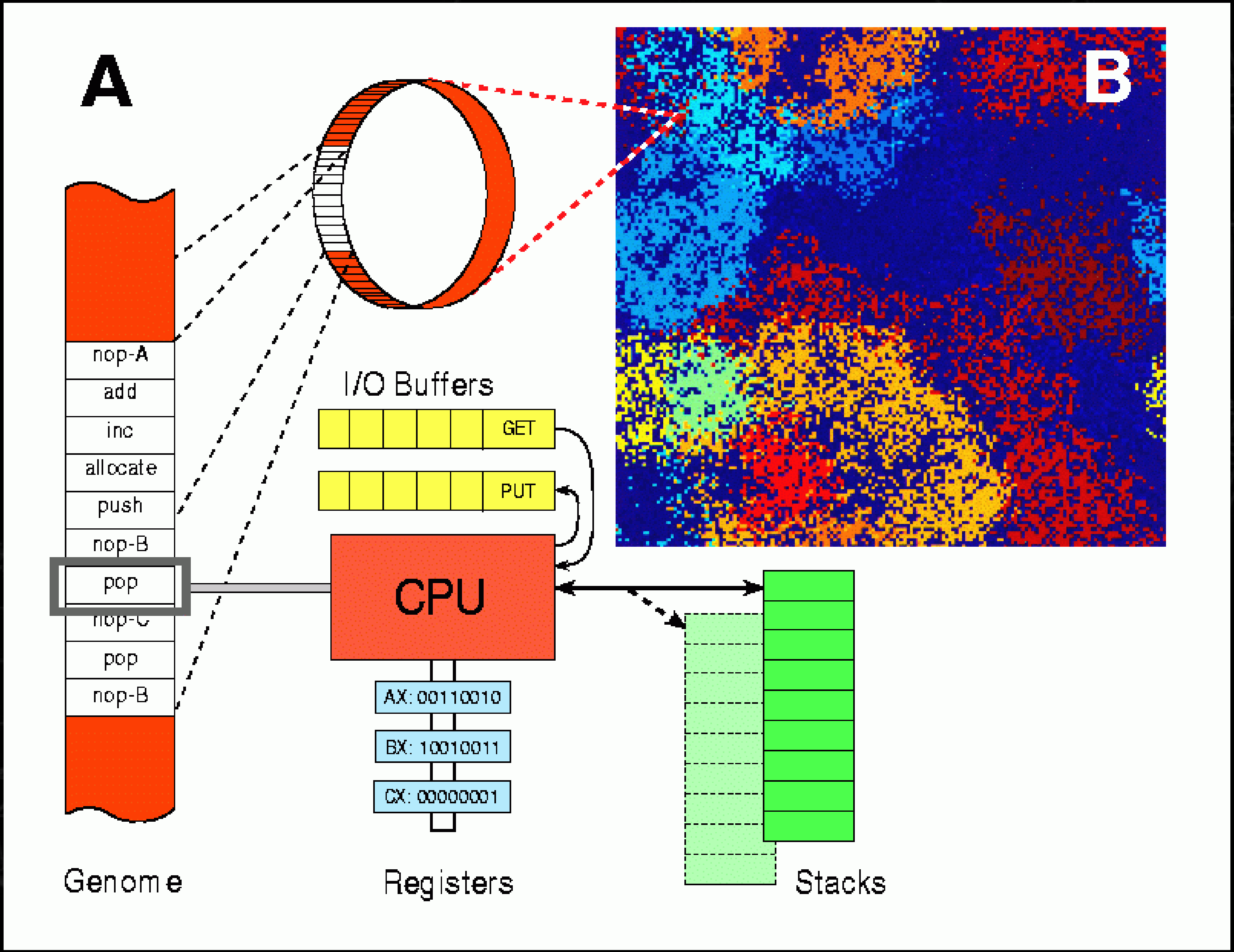
Developed at CalTech by C.Adami, C.Ofria, et al.

Avida: Genome



Adami, C. (2006)

Virtual World in Avida



Avida: Metabolism

- Main resource for programs is <u>CPU time</u> for which all programs compete
- Replication of digital information costs energy
- Energy is dispensed in units (quanta) that allow execution of a <u>single instruction</u> (single-instruction processing units)
- Digital organisms earn energy by:
  - Simply having long genomes
  - Performing logic operations two convert random input 32-bit numbers into random output 32-bit numbers

Avida: Metabolism

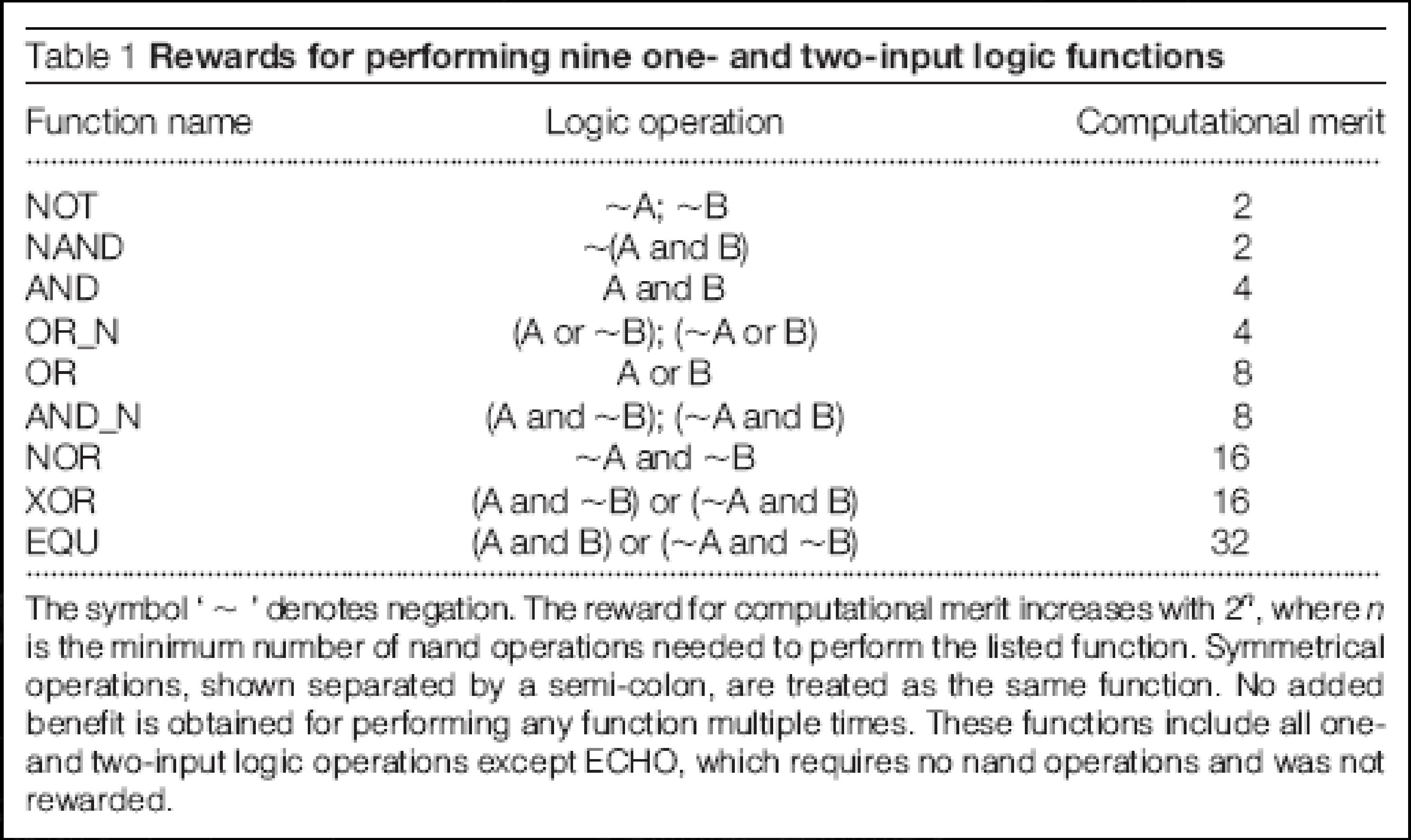


Table 1 **Rewards for performing nine one- and two-input logic functions**

Function name	Logic operation	Computational merit
NOT	~A; ~B	2
NAND	~(A and B)	2
AND	A and B	4
OR_N	(A or ~B); (~A or B)	4
OR	A or B	8
AND_N	(A and ~B); (~A and B)	8
NOR	~A and ~B	16
XOR	(A and ~B) or (~A and B)	16
EQU	(A and B) or (~A and ~B)	32

The symbol ' ~ ' denotes negation. The reward for computational merit increases with 2^n, where *n* is the minimum number of nand operations needed to perform the listed function. Symmetrical operations, shown separated by a semi-colon, are treated as the same function. No added benefit is obtained for performing any function multiple times. These functions include all one- and two-input logic operations except ECHO, which requires no nand operations and was not rewarded.

Lenski,R.E. et al. (2006)

Avida: Fitness

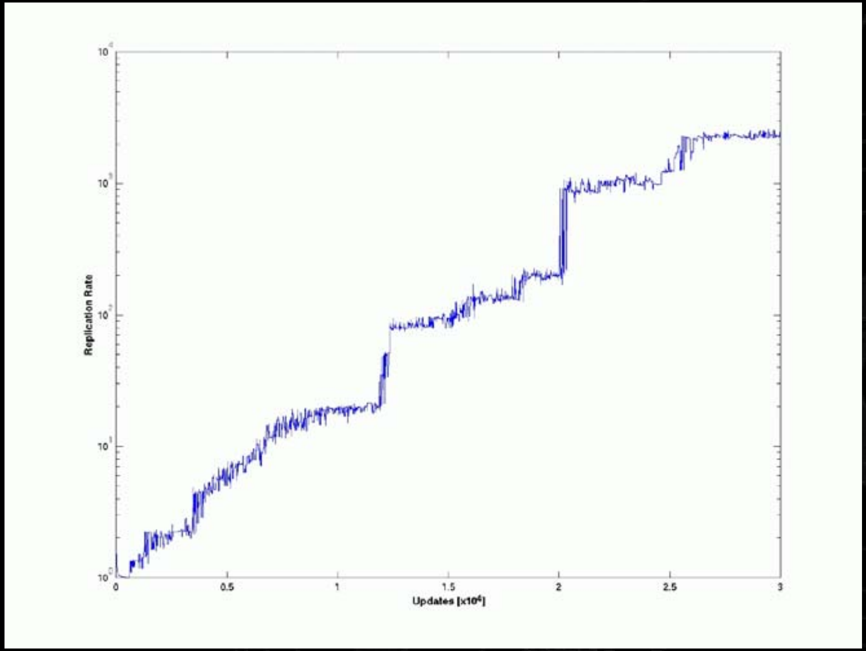
As for microbial and viral self-replicators, the fitness of a digital organism is given by the growth rate of the clone it gives rise to

The rate is a function of two factors:

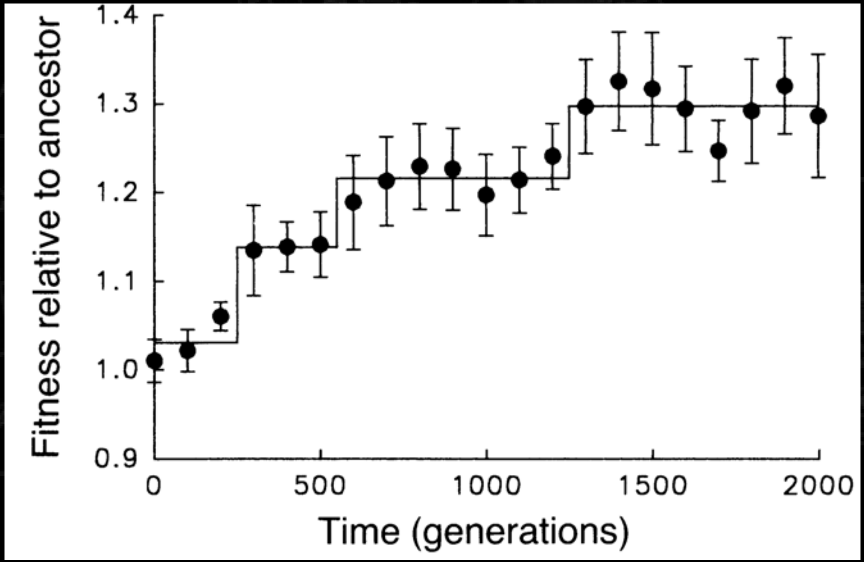
1) Efficiency of copy procedure
2) Ability to exploit environment to speed up replication

Limited resources (CPU time, space) force Darwinian evolution

Replication Rates

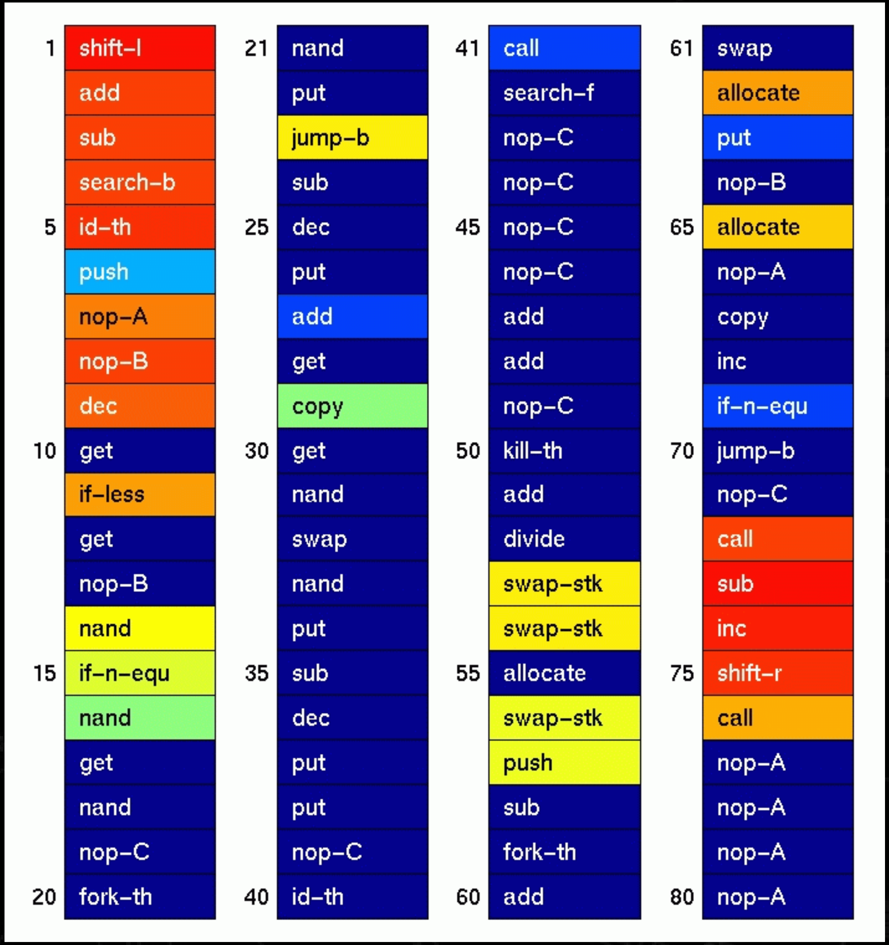


Avidians



E.Coli (MSU)

Typical Evolved Genome

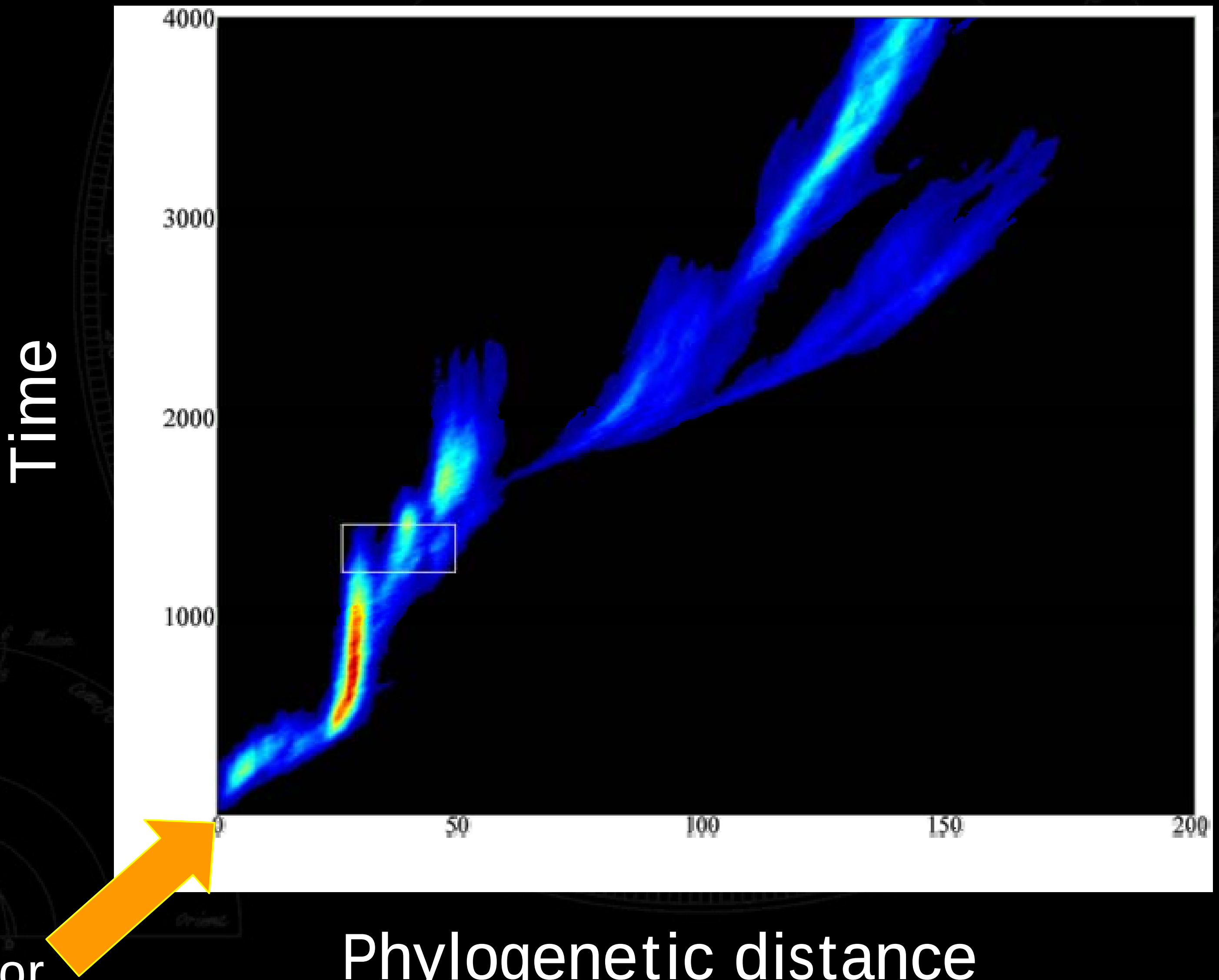


1	shift–l	21	nand	41	call	61	swap
	add		put		search–f		allocate
	sub		jump–b		nop–C		put
	search–b		sub		nop–C		nop–B
5	id–th	25	dec	45	nop–C	65	allocate
	push		put		nop–C		nop–A
	nop–A		add		add		copy
	nop–B		get		add		inc
	dec		copy		nop–C		if–n–equ
10	get	30	get	50	kill–th	70	jump–b
	if–less		nand		add		nop–C
	get		swap		divide		call
	nop–B		nand		swap–stk		sub
	nand		put		swap–stk		inc
15	if–n–equ	35	sub	55	allocate	75	shift–r
	nand		dec		swap–stk		call
	get		put		push		nop–A
	nand		put		sub		nop–A
	nop–C		nop–C		fork–th		nop–A
20	fork–th	40	id–th	60	add	80	nop–A

This genome has been color-coded to indicate the entropy of each code site

Red indicates variable (“hot”) sites, and blue the more conserved (“cold”) instructions

Phylogenetic Distance – Single Resource



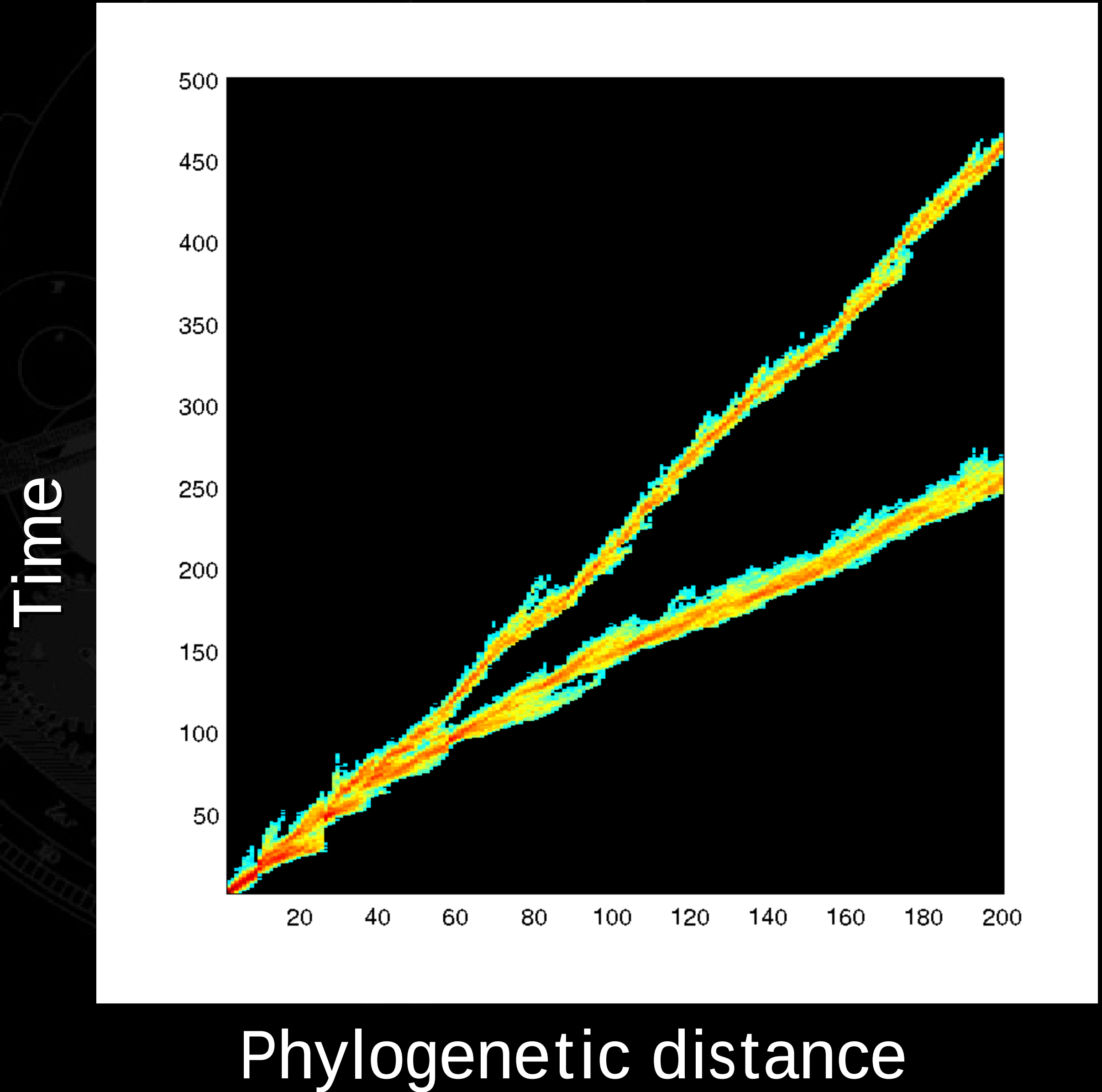
Phylogenetic distance

Phylogenetic Distance – Two Resources

- Up to now, environment for avidians has been simple: a single, homogenous and isotropic resource
- Resource diversity creates new level of complexity
- Multiple co-existing species can develop that adapt to using multiple resources, creating ecological networks

Phylogenetic Distance – Two Resources

Speciation



PolyWorld

Developed by Larry Yaeger

http://www.virgil.gr/11

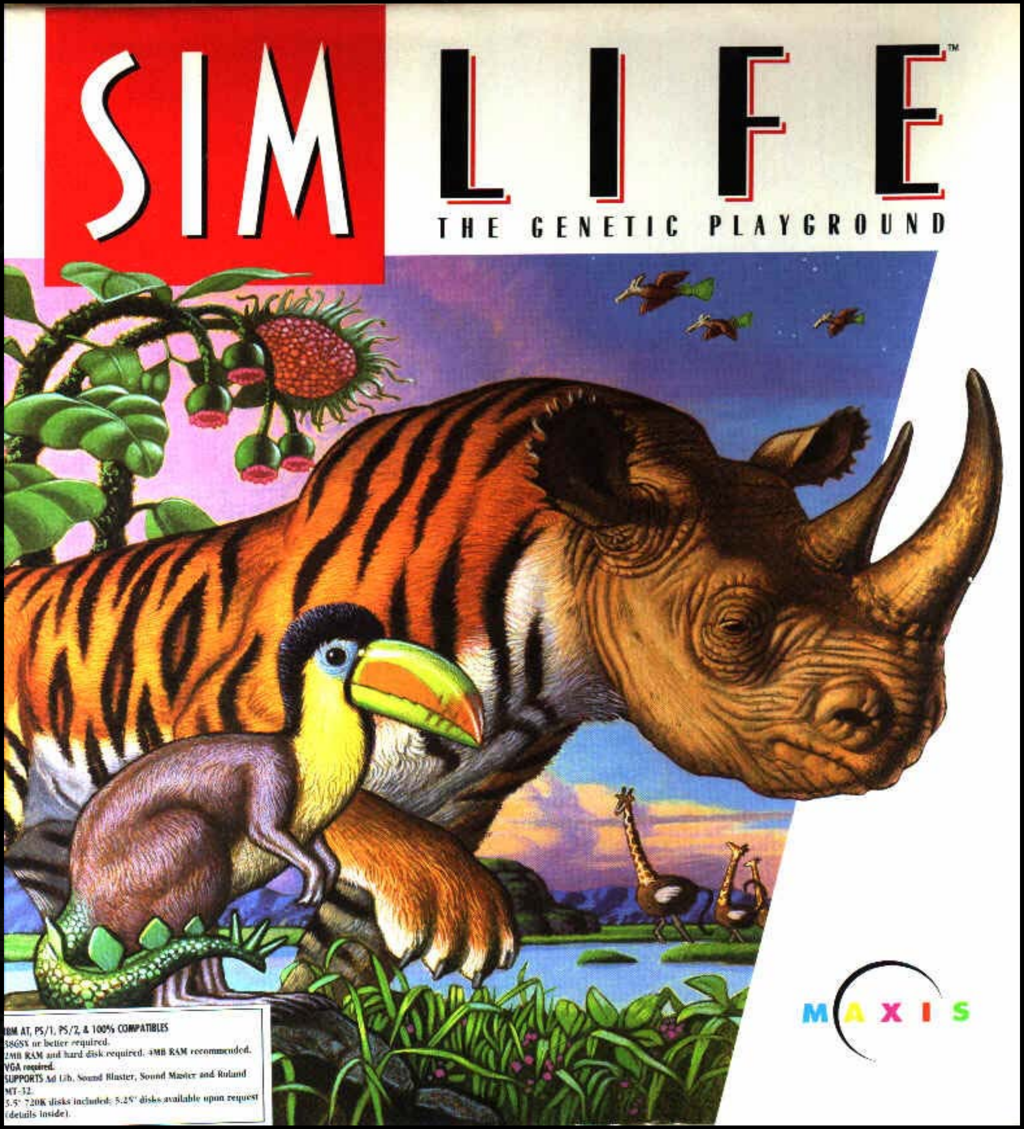
http://www.beanblossom.in.us/larryy/PolyWorld.html

PolyWorld: Example

- Brains
- Eating
- Killing and eating
- Swarming and foraging
- Mating

SimLife

- Computer game produced by Maxis in 1992
- Concept of the game is to simulate an ecosystem
- Players may modify the genetics of the plants and animals that inhabit the virtual world
- Point of game is to experiment and create a self sustaining ecosystem



IBM AT, PS/1, PS/2, & 100% COMPATIBLES
386SX or better required.
2MB RAM and hard disk required. 4MB RAM recommended.
VGA required.
SUPPORTS Ad Lib, Sound Blaster, Sound Master and Roland MT-32.
3.5" 720K disks included; 5.25" disks available upon request (details inside).

Open Challenges – Food For Thoughts

Why creature complexity has stopped increasing?

What’s limiting further development?

Over 10 years have passed:

- Memory space can support x10,000 bigger soup
- CPUs can crunch x100 faster
- In many cases “more is different” – is it here?