The evolutionary origin of complex features

Richard E. Lenski*, Charles Ofria†, Robert T. Pennock‡ & Christoph Adami§

** Department of Microbiology & Molecular Genetics, † Department of Computer Science & Engineering, and ‡ Lyman Briggs School & Department of Philosophy, Michigan State University, East Lansing, Michigan 48824, USA*
§ Digital Life Laboratory, California Institute of Technology, Pasadena, California 91125, USA

A long-standing challenge to evolutionary theory has been whether it can explain the origin of complex organismal features. We examined this issue using digital organisms—computer programs that self-replicate, mutate, compete and evolve. Populations of digital organisms often evolved the ability to perform complex logic functions requiring the coordinated execution of many genomic instructions. Complex functions evolved by building on simpler functions that had evolved earlier, provided that these were also selectively favoured. However, no particular intermediate stage was essential for evolving complex functions. The first genotypes able to perform complex functions differed from their non-performing parents by only one or two mutations, but differed from the ancestor by many mutations that were also crucial to the new functions. In some cases, mutations that were deleterious when they appeared served as stepping-stones in the evolution of complex features. These findings show how complex functions can originate by random mutation and natural selection.

Charles Darwin's theory of evolution[1], including its intertwined hypotheses of descent with modification and adaptation by natural selection, is widely regarded as one of the greatest scientific achievements of all time. From the outset, Darwin realized that "organs of extreme perfection and complication", such as the eye, posed a difficulty for his theory. Such features are much too complex to appear *de novo*, and he reasoned that they must evolve by incremental transitions through many intermediate states, sometimes undergoing changes in function. There now exists substantial evidence concerning the evolution of complex features that supports Darwin's general model[2–16]. Nonetheless, it is difficult to provide a complete account of the origin of any complex feature owing to the extinction of intermediate forms, imperfection of the fossil record, and incomplete knowledge of the genetic and developmental mechanisms that produce such features.

To examine the evolutionary origin of a complex feature in much greater detail than has previously been possible, we have performed experiments with digital organisms—computer programs that self-replicate, mutate and compete[17–26]. As Daniel Dennett[27] has emphasized, "…evolution will occur whenever and wherever three conditions are met: replication, variation (mutation), and differential fitness (competition)". By using this tractable system, we aim to shed light on principles relevant to any evolving system. Digital organisms are also of interest as computer scientists and engineers explore ways to apply evolutionary principles to program design, engineering and robotics[28–31]. The next section describes our experimental system, including the logic functions that digital organisms can use to obtain energy. There follows a case study of the genomic and phenotypic changes in a population that evolved an especially complex function, and then a functional-genomic analysis of the first genotype able to perform that function. Next we compare the trajectories and solutions of many populations that independently evolved this same function, and we conclude by examining the influence of different selective environments on the propensity of digital organisms to evolve the function.

Experimental system

Avida is a software platform for research on digital organisms[19]. To convey what digital organisms are, it is helpful to compare them with computer viruses. Digital organisms are self-replicating computer programs, as are viruses. However, computer viruses require direct intervention to mutate and evolve, whereas digital organisms exist in a computational environment where copying is imperfect such that they mutate randomly and evolve spontaneously. Also, digital organisms compete for energy and, depending on the environment, can obtain energy by performing logic functions[19,25].

In Avida, a genome is a circular sequence of instructions (Fig. 1) that are executed sequentially except when the execution of one instruction causes the instruction pointer to jump to another position. Jumps are encoded using a template-based system, which is more robust to mutation than is numerical addressing[17,24]. Each item in a sequence is one of 26 possible instructions (Supplementary Information). Reproduction is asexual and occurs by binary fission. Experiments began with an ancestor that could replicate but could not perform any logic functions. All organisms

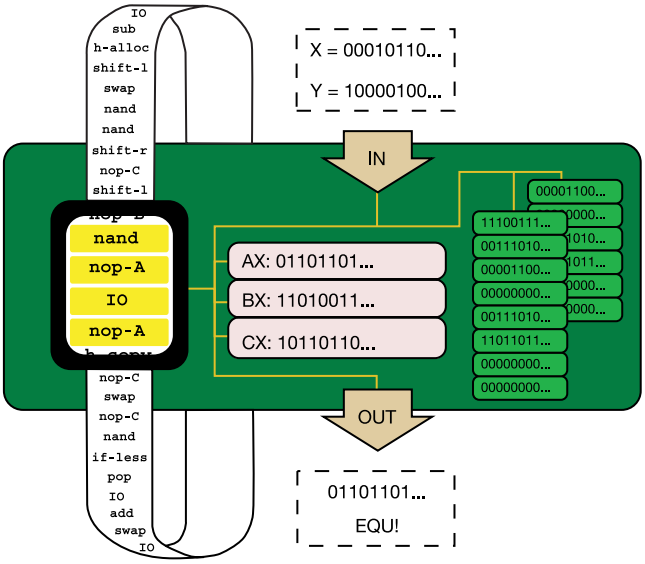


Figure 1 Digital organism in Avida. Each individual organism has a circular genome and a virtual CPU with two stacks and three registers that hold 32-bit strings. Execution of the genomic program generates a computational metabolism, whereby numerical substrates can be input from the environment, processed in stacks and registers, and resulting products output to the environment. See main text for details.

were identical and obtained equal energy to execute their genomic programs, including the copy commands by which a genome replicates itself one instruction at a time. Copying is subject to errors, including point mutations, insertions and deletions. Each mutation alters the genome and may change an organism's phenotype, including its replication efficiency, computational metabolism and robustness. Thus, genotypes vary in their expected reproductive success. As in nature, selection in Avida depends on the phenotypic effects of a mutation in its genetic context and in relation to the organism's environment; the researcher does not specify a distribution of selection coefficients. Most mutations in Avida are deleterious or neutral, but a small fraction increases fitness[20].

Digital organisms compete for the energy needed to execute instructions. Energy occurs as discrete quanta called 'single-instruction processing' units, or SIPs. Each SIP suffices to execute one instruction. By executing instructions, a digital organism can express phenotypes that enable it to obtain more energy and copy its genome. In Avida, organisms can acquire energy by two mechanisms. First, each organism receives SIPs in proportion to its genome length. Second, an organism can obtain further SIPs by performing one- and two-input logic operations on 32-bit strings (Supplementary Information). Only one of the 26 instructions in the genetic code, `nand` ('not and'), is itself a logic operator; and `nand` must be executed in coordination with `IO` ('input–output') instructions to perform the NAND function. All other logic functions can be constructed using one or more `nand` instructions within an integrated framework of other instructions.

Two logic functions, NAND and EQU ('equals'), are shown in Fig. 1. The execution of the highlighted `nand` instruction, when immediately followed by the modifying `nop-A` ('no operation') instruction, causes the bit-strings in the BX and CX registers to be combined according to the nand operator and the result to be written to the AX register. The nand operator returns 0 ('false') when both inputs are 1 ('true'); it returns 1 if one or both inputs are 0. The subsequent `IO` instruction and its modifying `nop-A` cause the string in the AX register to be output. If this output matches perfectly the correct answer for a function that is rewarded, then the organism's rate of energy acquisition, and hence the execution of its genomic program, is accelerated by the factor shown in Table 1. For example, if the organism in Fig. 1 had previously input the strings labelled X and Y, and if it now output the string in the AX register (generated by the preceding `nand` and `nop-A` instructions), then the organism would receive the reward for performing EQU. The reward is obtained because the output string has a 1 at every position where X and Y are equal (both 0 or both 1), and it has a 0 at each position where X and Y differ. The organism would obtain no reward if any of the 32 bits in the string were incorrect, or if it had already been rewarded in its lifetime for performing EQU.

Our study focuses on the origin of the EQU function. EQU garners the largest reward in our experiments because its performance is more complex than any other one- or two-input logic function, given the available genomic instructions. An exhaustive search shows that the minimum number of nand operations to perform EQU is five, which is greater than for any other one- or two-input logic function. Using the 26 available instructions, we wrote a program of length 19 that performs EQU but does not replicate (Supplementary Information). This program seems, but has not been proven, to be the shortest one to perform EQU.

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.

Case-study population

The ancestor could replicate but could not perform any logic function. However, an organism that evolved one or more of nine logic functions would obtain further energy. The benefit increased exponentially with the approximate difficulty of each function (Table 1). Functions could be performed in any order during an individual's life, but no extra energy was obtained by repeatedly performing the same function. No single mutation in the ancestor can produce even the simplest of these functions. Instead, several mutant instructions must appear in the same lineage, and such that they are coordinately executed, to perform even a simple function. Nonetheless, this population (and many others) evolved the capacity to perform EQU, the most complex of these functions.

Figure 2 shows the trajectory of the population's divergence from the ancestor. The vertical axis, phylogenetic depth, is the cumulative number of generations in which an individual differed from its parent by one or more mutations. The horizontal axis is time in computational updates (see Methods). The colour scale shows the abundance of genotypes at a given depth. The blue line shows the exact line of descent leading to the genotype that was most abundant at the end of the experiment. This final dominant type was 344 steps removed from its ancestor. That is, an offspring differed genetically from its parent in 344 of the many thousands of generations leading to this final type. The final dominant has a genome 83 instructions long (the ancestral length was 50), and it performs all nine logic functions that provide energy.

Figure 3 shows the trajectories for two fitness components, replication efficiency and computational merit, for all 344 geno-

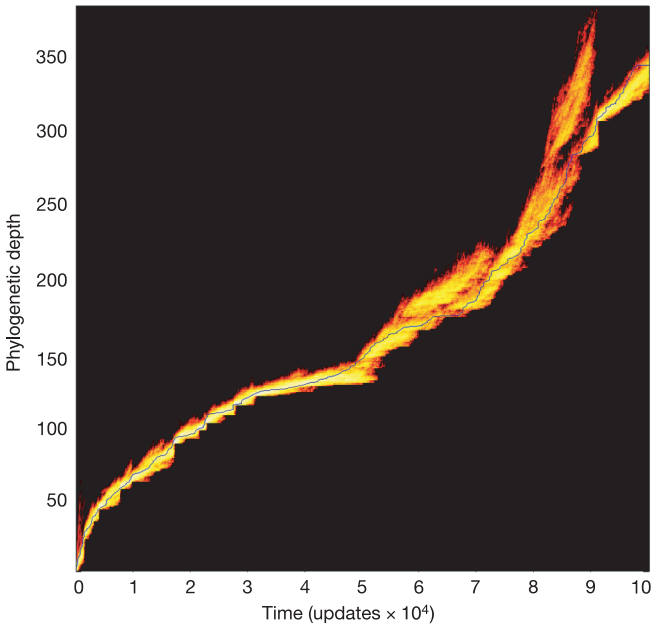


Figure 2 Phylogenetic depth versus time in the case-study population. Phylogenetic depth is the cumulative number of generations in which an organism's genotype differs from its parent. The exact line of descent leading to the most abundant final genotype is shown as the blue line. Colours indicate the relative abundance of genotypes at any depth, yellow being more abundant than red.

types along the line of descent. Replication efficiency is the ratio of an organism's genome length to the SIPs used during its life cycle. Computational merit is the total reward obtained over an organism's lifetime for performing logic functions. Each genotype's expected replication rate, or fitness, equals the product of these quantities. We also identified every mutation that fell along the line of descent in this population, and characterized the phenotypic changes associated with each step (Supplementary Information). The EQU function first appeared at step 111 (update 27,450). There were 103 single mutations, six double mutations, and two triple mutations among these steps. Forty-five of the steps increased overall fitness, 48 were neutral and 18 were deleterious relative to the immediate parent. The large proportion of beneficial mutations along the line of descent is not surprising, because this lineage represents the eventual winners that were assembled by natural selection. Thirteen of the 45 beneficial steps gave rise to the expression of logic functions not expressed by the immediate parent, including six steps in which there were trade-offs of simpler for more complex functions.

The presence of deleterious mutations along the line of descent is more surprising. Fifteen of the 18 deleterious mutations reduced fitness by $<3\%$ relative to the parent, and might have hitchhiked with beneficial mutations that arose soon after in the same genetic background. However, two mutations reduced fitness by $>50\%$. One was a point mutation that disrupted replication efficiency. Its harmful effect was eliminated by the next mutation in the line of descent, which occurred at a distant site in the genome. The other very deleterious step was a point mutation, at depth 110, that knocked out NAND, one of the simplest logic functions. Only two individuals had this maladapted genotype, yet their descendants emerged as eventual winners. In fact, in the very next step, this genotype produced the mutation that gave rise to EQU. Was that deleterious mutation extremely lucky to hitchhike with such a beneficial mutation? Or was the deleterious mutation a prerequisite for producing the EQU function within that genome context? To distinguish between these hypotheses, we reversed this one-step-prior mutation in the genotype that first expressed EQU. This reversal eliminated the EQU function. Therefore, a mutation that was highly deleterious when it appeared was highly beneficial in combination with a subsequent mutation. The evolution of a complex feature, such as EQU, is not always an inexorably upward climb toward a fitness peak, but instead may involve sideways and even backward steps, some of which are important.

After the origin of EQU, another 233 steps occurred along the line of descent leading to the final dominant genotype. Of these, 62 were beneficial, 132 neutral and 39 deleterious. All nine logic functions were performed from genotype 306 onwards.

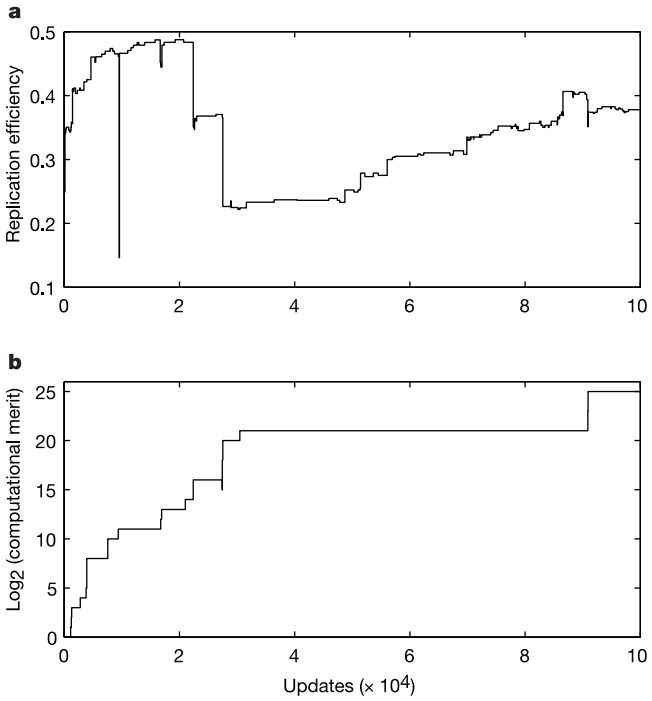


Figure 3 Trajectories for two fitness components, showing each genotype in the line of descent for the case-study population. **a**, Replication efficiency, which is the ratio of genome length to energy used in an organism's lifetime. **b**, Computational merit, shown $\log_2$-transformed, which is the product of all the rewards obtained by an organism for logic functions performed during its lifetime.

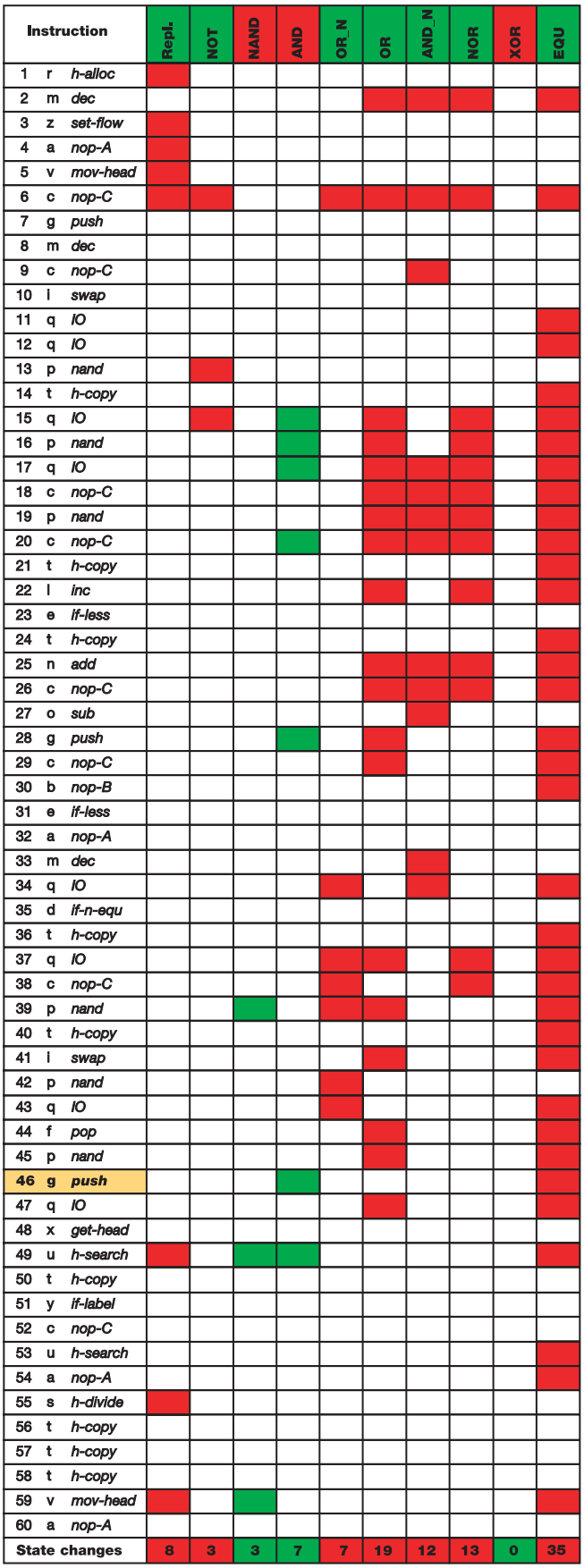
Functional genomics

To determine how many instructions were required to perform EQU at its origin in the case-study population, we systematically replaced each existing instruction with a null instruction in the first genotype able to perform EQU. We scored each null mutant for which functions were lost and which remained, and the data from all the mutants were combined to produce an array showing the relationship between genome sequence and phenotypic properties (Fig. 4). The genome of the first EQU-performing organism had 60 instructions; eliminating any of 35 of them destroyed that function. Although the mutation of only one instruction produced this innovation when it originated, the EQU function evidently depends on many interacting components.

The many instructions required to perform EQU, the previous evolution of simpler functions, and the repeated trade-offs between simpler and more complex functions all suggest functional interconnections between the genetic networks encoding them. We used the same array to ask how many instructions that were required to perform EQU were also required for other functions. Besides EQU, this genotype performed five of the eight simpler logic functions; AND was lost as a side-effect of the EQU-producing mutation, and NAND had been eliminated by the one-step-prior mutation. Of the 35 instructions required for EQU, 22 were needed for simpler functions. Three instructions required for EQU were also essential for replication; these were conserved from the ancestral sequence, as were five others. However, 27 of the 35 instructions required to perform EQU were evolutionarily derived, and all but one of them had appeared in the line of descent before this function was ever performed. Thus, although more than two dozen mutations were used to build EQU, undoing any one of them destroyed this function. This asymmetry might suggest that EQU is fragile and cannot last. However, the EQU function lasted throughout the experiment because, once evolved, it was so valuable that defective mutants were eliminated by selection.

Variations on a theme

The case-study population was one of 50 that evolved under identical conditions, 23 of which acquired EQU. The phylogenetic depth at which EQU first appeared ranged from 51 to 721 steps. In principle, 16 mutations, coupled with three instructions already present in the ancestor, could have produced an EQU-performing organism. The actual paths were much longer and highly variable, indicating the circuitousness and unpredictability of evolution leading to this complex feature. We identified the pivotal genotypes that first performed EQU in each population, and the pivotal mutations that distinguished them from their parents. We use 'pivotal' to mark these milestones, not to imply that earlier genotypes and mutations were unimportant for the origin of EQU. A single mutation distinguished the pivotal genotype from its parent in 19 populations, whereas four involved double mutations. The pivotal mutations included point mutations, insertions, a small duplication and even deletions. Pivotal point



Instruction			Repl.	NOT	NAND	AND	OR_N	OR	AND_N	NOR	XOR	EQU
1	r	*h-alloc*										
2	m	*dec*										
3	z	*set-flow*										
4	a	*nop-A*										
5	v	*mov-head*										
6	c	*nop-C*										
7	g	*push*										
8	m	*dec*										
9	c	*nop-C*										
10	l	*swap*										
11	q	*IO*										
12	q	*IO*										
13	p	*nand*										
14	t	*h-copy*										
15	q	*IO*										
16	p	*nand*										
17	q	*IO*										
18	c	*nop-C*										
19	p	*nand*										
20	c	*nop-C*										
21	t	*h-copy*										
22	l	*inc*										
23	e	*if-less*										
24	t	*h-copy*										
25	n	*add*										
26	c	*nop-C*										
27	o	*sub*										
28	g	*push*										
29	c	*nop-C*										
30	b	*nop-B*										
31	e	*if-less*										
32	a	*nop-A*										
33	m	*dec*										
34	q	*IO*										
35	d	*if-n-equ*										
36	t	*h-copy*										
37	q	*IO*										
38	c	*nop-C*										
39	p	*nand*										
40	t	*h-copy*										
41	l	*swap*										
42	p	*nand*										
43	q	*IO*										
44	f	*pop*										
45	p	*nand*										
46	g	*push*										
47	q	*IO*										
48	x	*get-head*										
49	u	*h-search*										
50	t	*h-copy*										
51	y	*if-label*										
52	c	*nop-C*										
53	u	*h-search*										
54	a	*nop-A*										
55	s	*h-divide*										
56	t	*h-copy*										
57	t	*h-copy*										
58	t	*h-copy*										
59	v	*mov-head*										
60	a	*nop-A*										
State changes			8	3	3	7	7	19	12	13	0	35

mutations used 10 of the 26 possible instructions; and several pivotal insertions and deletions were unique. Therefore, many different mutations produced the EQU function in the right genomic context.

Before the pivotal mutations, each line of descent had already experienced 50 or more steps, including beneficial, neutral and deleterious mutations. Some deleterious mutations had small effects and hitchhiked with beneficial mutations; others reverted or were compensated by mutations elsewhere. But in the case study, we showed that one deleterious mutation had genetically predisposed (through an epistatic interaction) the subsequent origin of EQU. To investigate this possibility further, we examined all of the mutations one step before the pivotal mutations. Five of the 23 one-step-prior mutations were deleterious in the backgrounds in which they occurred (including the case study). When these five were reverted in the pivotal genotypes, the EQU function was eliminated in three cases. Thus, three genotypes that first performed this complex function depended on a one-step-prior mutation that was deleterious when it appeared.

The 23 pivotal genomes ranged in length from 49 to 356 instructions; the ancestor had 50 instructions and so there was a strong tendency for increased length to precede the origin of EQU. The parents of the pivotal genotypes performed between four and all eight of the simpler logic functions, with a median of seven. Therefore, at least several simpler functions evolved before EQU in every population. However, no particular function was essential for the subsequent evolution of EQU, because each one was absent from at least one of the 23 parent genotypes. All the pivotal mutations were beneficial owing to the extra energy obtained by the EQU function. However, net gains also depended on pleiotropic side-effects on other traits. Twenty of the 23 pivotal mutations caused losses of one or more logic functions the parent performed, and 13 reduced replication efficiency. Eight yielded another function, and three improved replication efficiency. On balance, 20 of the 23 pivotal mutations would have been deleterious if EQU had not been rewarded, whereas three would still have been beneficial. Thus, beneficial mutations that produced this new function usually, but not always, engendered trade-offs in other aspects of performance.

We ran the functional-genomic analyses on all 23 pivotal genotypes. The number of instructions required for EQU ranged from 17 to 43, with a median of 28 instructions. Notice that one evolved type apparently needed only 17 instructions to perform EQU, whereas our shortest hand-written program used 19 instructions. Further examination showed that this unexpectedly low value occurred because the evolved genome was partially redundant, such that one-at-a-time null mutations did not reveal the full extent of its computational network. These analyses therefore provide a minimum estimate of the number of instructions that a genotype uses to perform a function.

Different environments

We carried out experiments to examine the effects of different selective environments on the propensity to evolve the EQU function, with all other conditions held constant. Ten further populations evolved under each of 36 possible regimes in which one or

Figure 4 Functional-genomic array for the first organism to perform EQU in the case-study population. Its genome sequence is shown to the left; the instruction highlighted in yellow is the pivotal mutation that yielded EQU but simultaneously eliminated AND. Top labels denote replication (Repl.) and logic functions; associated colours show whether this organism can (green) or cannot (red) perform the function. The fill in each interior cell shows the effect on the function of replacing the instruction with a null instruction. Red, null mutation destroys existing function; blank, null mutation has no qualitative effect; green, null mutation produces new function. The number of state changes for each function is shown at the bottom.

two simpler functions were not rewarded. In all environments, at least one population evolved EQU. Evidently, neither any particular simpler function nor any pairwise combination of functions was required to evolve this complex feature. In these 36 environments, the overall fraction of populations that evolved EQU was 124 of 360 (34%), only slightly less than in the 'reward-all' environment ($P = 0.0764$, one-tailed Fisher's exact test).

At the other extreme, 50 populations evolved in an environment where only EQU was rewarded, and no simpler function yielded energy. We expected that EQU would evolve much less often because selection would not preserve the simpler functions that provide foundations to build more complex features. Indeed, none of these populations evolved EQU, a highly significant difference from the fraction that did so in the reward-all environment ($P \approx 4.3 \times 10^{-9}$, Fisher's exact test). However, these populations tested more genotypes, on average, than did those in the reward-all environment (2.15×10^7 versus 1.22×10^7; $P < 0.0001$, Mann-Whitney test), because they tended to have smaller genomes, faster generations, and thus turn over more quickly. However, all populations explored only a tiny fraction of the total genotypic space. Given the ancestral genome of length 50 and 26 possible instructions at each site, there are $\sim 5.6 \times 10^{70}$ genotypes; and even this number underestimates the genotypic space because length evolves.

Discussion

By using digital organisms, we traced the exact genealogy, without any 'missing links', from an ancestor that could replicate only to descendants able to perform multiple logic functions requiring the coordinated execution of many genomic instructions. The most complex function, EQU, evolved only when several simpler functions were also useful. Some simpler functions were accessible from the ancestor by relatively few mutations, and these served as a foundation on which more complex features were built. The foundational role of simpler functions in the origin of more complex ones was evident in the overlap of the genetic networks underlying their expression, and the frequent loss of simpler functions as side-effects of mutations yielding more complex functions. Our experiments demonstrate the validity of the hypothesis, first articulated by Darwin[1] and supported today by comparative and experimental evidence[2–16], that complex features generally evolve by modifying existing structures and functions. Some readers might suggest that we 'stacked the deck' by studying the evolution of a complex feature that could be built on simpler functions that were also useful. However, that is precisely what evolutionary theory requires, and indeed, our experiments showed that the complex feature never evolved when simpler functions were not rewarded. Our experiments also show that many different genomic solutions produce the same complex function. Following any particular path is extremely unlikely, but the complex function evolved with a high probability, implying a very large number of potential paths[32]. Although the complex feature first appeared as the immediate result of only one or two mutations, its function invariably depended on many instructions that had previously evolved to perform other functions, such that their removal would eliminate the new feature.

Of course, digital organisms differ from organic life in their genetic constitution, metabolic activities and physical environments. However, digital organisms undergo the same processes of reproduction, mutation, inheritance and competition that allow evolution and adaptation by natural selection in organic forms. Other similarities are pleiotropy (one mutation affects multiple traits) and epistasis (multiple mutations interact to determine the same trait), which emerge in both organic and digital life from the nonlinear processes by which genomes encode and produce phenotypic features. As a consequence, selection acts on organisms rather than directly on their genes. The organisms in our study were asexual. Sex might accelerate the evolution of complex features by combining functions that independently evolved in different lineages. On the other hand, asexuality permits beneficial combinations of mutations to spread even when they are individually deleterious, as sometimes occurred in our experiments. The consequences of asexual and sexual reproduction for the evolution of complex features deserve further research. In closing, digital organisms provide opportunities to address important issues in evolutionary biology. They are particularly well suited to problems that are difficult to study with organic forms owing to incomplete information, insufficient time and the impracticality of experiments. □

Methods

Avida software

Avida uses a time-slicing algorithm[19] to ensure that all organisms execute instructions in an effectively parallel manner. Experiments ran using Avida version 1.6 on the Linux operating system on a Beowulf cluster of 64 Pentium III processors. The software and configuration files for our experiments can be obtained free from our website (http://myxo.css.msu.edu/papers/nature2003). More information about Avida, and further data from our experiments, can also be found there.

Experimental conditions

Every population started with 3,600 identical copies of an ancestral genotype that could replicate but could not perform any logic functions. Each replicate population that evolved in the same environment was seeded with a different random number. The hand-written ancestral genome was 50 instructions long, of which 15 were required for efficient self-replication; the other 35 were tandem copies of a single no-operation instruction (`nop-C`) that performed no function when executed. Copy errors caused point mutations, in which an existing instruction was replaced by any other (all with equal probability), at a rate of 0.0025 errors per instruction copied. Single-instruction deletions and insertions also occurred, each with a probability of 0.05 per genome copied. Hence, in the ancestral genome of length 50, 0.225 mutations are expected, on average, per replication. Various organisms from nature have genomic mutation rates higher or lower than this value[33]. Mutations in Avida also occasionally cause the asymmetrical division of a copied genome, leading to the deletion or duplication of multiple instructions. Each digital organism obtained 'energy' in the form of SIPs at a relative rate (standardized by the total demand of all organisms in the population) equal to the product of its genome length and computational merit, where the latter is the product of rewards for logic functions performed. The exponential reward structure shown in Table 1 was used in the reward-all environment, whereas some functions obtained no reward under other regimes. An organism's expected reproductive rate, or fitness, equals its rate of energy acquisition divided by the amount of energy needed to reproduce. Fitness can also be decomposed into replication efficiency (ratio of genome length to energy required for replication) and computational merit. Each population evolved for 100,000 updates, an arbitrary time unit equal to the execution of 30 instructions, on average, per organism. The ancestor used 189 SIPs to produce an offspring, so each run lasted for 15,873 ancestral generations. Populations existed on a lattice with a capacity of 3,600 individuals. When an organism copied its genome and divided, the resulting offspring was randomly placed in one of the eight adjacent cells or in the parent's cell. Each birth caused the death of the individual that was replaced, thus maintaining a constant population size.

Received 19 September 2002; accepted 13 March 2003; doi:10.1038/nature01568.

1. Darwin, C. *On the Origin of Species by Means of Natural Selection* (Murray, London, 1859).
2. Jacob, F. Evolution and tinkering. *Science* **196,** 1161–1166 (1977).
3. Salvini-Plawen, L. V. & Mayr, E. On the evolution of photoreceptors and eyes. *Evol. Biol.* **10,** 207–263 (1977).
4. Mortlock, R. R. (ed.) *Microorganisms as Model Systems for Studying Evolution* (Plenum, New York, 1984).
5. Dawkins, R. *The Blind Watchmaker* (Norton, New York, 1986).
6. Goldsmith, T. H. Optimization, constraint, and history in the evolution of eyes. *Q. Rev. Biol.* **65,** 281–322 (1990).
7. Piatigorsky, J. & Wistow, G. The recruitment of crystallins: new functions precede gene duplication. *Science* **252,** 1078–1079 (1991).
8. Land, M. F. & Fernald, R. D. The evolution of eyes. *Annu. Rev. Neurosci.* **15,** 1–29 (1992).
9. Nilsson, D.-E. & Pelger, S. A pessimistic estimate of the time required for an eye to evolve. *Proc. R. Soc. Lond. B* **256,** 53–58 (1994).
10. Meléndez-Hevia, E., Waddell, T. G. & Cascante, M. The puzzle of the Krebs citric acid cycle: assembling the pieces of chemically feasible reactions, and opportunism in the design of metabolic pathways during evolution. *J. Mol. Evol.* **43,** 293–303 (1996).
11. Chen, L., DeVries, A. L. & Cheng, C.-H. C. Evolution of antifreeze glycoprotein gene from a trypsinogen gene in Antarctic notothenioid fish. *Proc. Natl Acad. Sci. USA* **94,** 3811–3816 (1997).
12. Dean, A. M. & Golding, G. B. Protein engineering reveals ancient adaptive replacements in isocitrate dehydrogenase. *Proc. Natl Acad. Sci. USA* **94,** 3104–3109 (1997).
13. Newcomb, R. D. *et al.* A single amino acid substitution converts a carboxylesterase to an organophosphorus hydrolase and confers insecticide resistance on a blowfly. *Proc. Natl Acad. Sci. USA* **94,** 7464–7468 (1997).
14. Miller, K. R. *Finding Darwin's God* (Cliff Street, New York, 1999).
15. Yokoyama, S. & Radlwimmer, F. B. The molecular genetics and evolution of red and green colour vision in vertebrates. *Genetics* **158,** 1697–1710 (2001).
16. Wilkins, A. S. *The Evolution of Developmental Pathways* (Sinauer, Sunderland, Massachusetts, 2002).

17. Ray, T. S. in *Artificial Life II* (eds Langton, C. G., Taylor, C., Farmer, J. D. & Rasmussen, S.) 371–408 (Addison–Wesley, Redwood City, California, 1991).
18. Bedau, M. A., Snyder, E., Brown, C. T. & Packard, N. H. in *Proc. 4th Eur. Conf. Artif. Life* (eds Husband, P. & Harvey, I.) 125–134 (MIT Press, Cambridge, Massachusetts, 1997).
19. Adami, C. *Introduction to Artificial Life* (Springer, New York, 1998).
20. Lenski, R. E., Ofria, C., Collier, T. C. & Adami, C. Genome complexity, robustness and genetic interactions in digital organisms. *Nature* **400,** 661–664 (1999).
21. Wagenaar, D. & Adami, C. in *Artificial Life VII* (eds Bedau, M. A., McCaskill, J. S., Packard, N. H. & Rasmussen, S.) 216–220 (MIT Press, Cambridge, Massachusetts, 2000).
22. Wilke, C. O., Wang, J. L., Ofria, C., Lenski, R. E. & Adami, C. Evolution of digital organisms at high mutation rate leads to survival of the flattest. *Nature* **412,** 331–333 (2001).
23. Yedid, G. & Bell, G. Microevolution in an electronic microcosm. *Am. Nat.* **157,** 465–487 (2001).
24. Ofria, C., Adami, C. & Collier, T. C. Design of evolvable computer languages. *IEEE Trans. Evol. Comput.* **6,** 420–424 (2002).
25. Wilke, C. O. & Adami, C. The biology of digital organisms. *Trends Ecol. Evol.* **17,** 528–532 (2002).
26. Yedid, G. & Bell, G. Macroevolution simulated with autonomously replicating computer programs. *Nature* **420,** 810–812 (2002).
27. Dennett, D. in *Encyclopedia of Evolution* (ed. Pagel, M.) E83–E92 (Oxford Univ. Press, New York, 2002).
28. Holland, J. H. *Adaptation in Natural and Artificial Systems* (MIT Press, Cambridge, Massachusetts, 1992).
29. Koza, J. R. *Genetic Programming* (MIT Press, Cambridge, Massachusetts, 1992).
30. Lipson, H. & Pollack, J. B. Automatic design and manufacture of robotic life forms. *Nature* **406,** 974–978 (2000).
31. Foster, J. A. Evolutionary computation. *Nature Rev. Genet.* **2,** 428–436 (2001).
32. Dennett, D. C. *Darwin's Dangerous Idea* (Simon & Schuster, New York, 1995).
33. Drake, J. W., Charlesworth, B., Charlesworth, D. & Crow, J. F. Rates of spontaneous mutation. *Genetics* **148,** 1667–1686 (1998).

Supplementary Information accompanies the paper on **www.nature.com/nature**.

Acknowledgements We thank A. Bennett, J. Bull, J. Coyne, D. Lenski, M. Lenski and E. Zuckerkandl for comments. The authors' work is supported by the US National Science Foundation Biocomplexity Program and by the MSU Foundation. Part of this work was carried out at the Jet Propulsion Laboratory under contract with the US National Aeronautics and Space Administration.

Competing interests statement The authors declare that they have no competing financial interests.

Correspondence and requests for materials should be addressed to R.E.L. (lenski@msu.edu).