Proc. Natl. Acad. Sci. USA
Vol. 92, pp. 10742–10746, November 1995
Computer Sciences

The evolution of emergent computation

JAMES P. CRUTCHFIELD*† AND MELANIE MITCHELL‡

*Physics Department, University of California, Berkeley, CA 94720; and ‡Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501

Communicated by Murray Gell-Mann, Santa Fe Institute, Santa Fe, NM, August 15, 1995

ABSTRACT **A simple evolutionary process can discover sophisticated methods for emergent information processing in decentralized spatially extended systems. The mechanisms underlying the resulting emergent computation are explicated by a technique for analyzing particle-based logic embedded in pattern-forming systems. Understanding how globally coordinated computation can emerge in evolution is relevant both for the scientific understanding of natural information processing and for engineering new forms of parallel computing systems.**

Many systems in nature exhibit sophisticated collective information-processing abilities that emerge from the individual actions of simple components interacting via restricted communication pathways. Some often-cited examples include efficient foraging and intricate nest-building in insect societies (1), the spontaneous aggregation of a reproductive multicellular organism from individual amoeba in the life cycle of the *Dictyostelium* slime mold (2), the parallel and distributed processing of sensory information by assemblies of neurons in the brain (3), and the optimal pricing of goods in an economy arising from agents obeying local rules of commerce (4). Allowing global coordination to emerge from a decentralized collection of simple components has important advantages over explicit central control in both natural and human-constructed information-processing systems. There are substantial costs incurred in having centralized coordination, not the least being (*i*) speed (a central coordinator can be a bottleneck to fast information processing), (*ii*) robustness (if the central coordinator is injured or lost, the entire system collapses), and (*iii*) equitable resource allocation (a central controller must be allocated a lion's share of system resources that otherwise could go to other agents in the system) (e.g., see ref. 5). However, it is difficult to design a collection of individual components and their local interactions in a way that will give rise to useful global information processing. It is not well understood how such apparent complex global coordination emerges from simple individual actions in natural systems or how such systems are produced by biological evolution. This paper reports the application of new methods for detecting computation in nonlinear processes to a simple evolutionary model that allows us to address these questions directly. The main result is the evolutionary discovery of methods for emergent global computation in a spatially distributed system consisting of locally interacting processors.

We use the general term "emergent computation" to describe the appearance of global information processing in such systems (see refs. 6 and 7). Our goal is to understand the mechanisms by which evolution can discover methods of emergent computation. We are studying this question in a theoretical framework that, while simplified, still captures the essence of the phenomena of interest. This framework requires (*i*) an idealized class of decentralized system in which global information processing can arise from the actions of simple, locally connected units; (*ii*) a computational task that necessitates global information processing; and (*iii*) an idealized computational model of evolution.

One of the simplest systems in which emergent computation can be studied is a one-dimensional binary-state cellular automaton (CA) (8)—a one-dimensional spatial lattice of N identical two-state machines ("cells"), each of which changes its state as a function only of the current states in a local neighborhood of radius r. The lattice starts out with an initial configuration (IC) of N cell states (0s and 1s). This configuration changes in discrete time steps according to the CA "rule"—a look-up table mapping neighborhood state configurations to update states. At each time step, all cells examine their local neighborhoods (subject to specified boundary conditions), consult the look-up table, and update their states simultaneously. The CA's radius places an upper boundary on the speed of information transmission through the lattice. It also limits the sophistication of the local dynamics: the number of look-up table entries is 2^{2r+1}. Thus, fixing $r \ll N$ constrains the sophistication of a CA's explicit information processing.

A simple-to-define computational task for CAs that requires global information processing is deciding whether or not the IC contains more than half 1s. We call this the $\rho_c = 1/2$ task, with ρ_c denoting a threshold density of 1s in the input. If ρ_0 denotes the density of 1s in the IC, the desired behavior is for all cells to quickly change to state 1 if $\rho_0 > \rho_c$ and to quickly change to state 0 if $\rho_0 < \rho_c$. The $\rho_c = 1/2$ task requires global communication, since ρ_0 is a global property of the entire lattice; no linear combination of local computations—such as the cells computing the majority of 1s in their neighborhood—can solve this problem. Designing an algorithm to perform the $\rho_c = 1/2$ task is trivial for systems with a central controller of some kind, such as a standard computer with a counter register or a neural network with global connectivity. But it is difficult to design a decentralized, spatially extended system such as a CA to perform this task, since there is no central counter or global communication built in. It can be shown that no finite-radius CA can perform this task perfectly across all lattice sizes (9, 10), but even to perform this task well for a fixed lattice size requires more powerful computation than can be performed by a single cell or any linear combination of cells. Since the 1s can be distributed throughout the CA lattice, the CA must transfer information over large space-time distances ($\approx N$), and information from distant parts of the lattice must interact so as to perform the computation. With $r \ll N$, such information transmission and interaction can be accomplished only through the coordination of emergent high-level signals. Thus, this task is well suited for investigating the ability of an evolutionary process to design CAs with sophisticated emergent computational abilities.

One class of computational models of evolution are genetic algorithms (GAs) (11), which evolve a population of candidate solutions to an optimization problem by propagating the most "fit" candidates to the next generation via genetic modifications. We carried out a set of experiments in which a GA was used to evolve one-dimensional binary-state $r = 3$ CAs (with

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. §1734 solely to indicate this fact.

Abbreviations: CA, cellular automaton; IC, initial configuration; GA, genetic algorithm.

†To whom reprint requests should be addressed.

spatially periodic boundary conditions) to perform the $\rho_c = 1/2$ task. This GA, while highly idealized, contained the rudiments of selection and variation: crossover and mutation worked on the genotype (the 128-bit string encoding the CA look-up table), whereas selection was according to the fitness of the phenotype (the CA's spatiotemporal behavior on an $N = 149$ cell lattice). The GA started out with an initial population of 100 strings ("rules") randomly generated with a uniform distribution over the fraction of 1s in the string. The "fitness" of each rule was computed by iterating the corresponding CA on 100 randomly chosen ICs uniformly distributed over $\rho_0 \in [0, 1]$, half with $\rho_0 < \rho_c$ (correct classification: all 0s) and half with $\rho_0 > \rho_c$ (correct classification: all 1s), and by recording the fraction of correct classifications performed in a maximum of slightly more than $2N$ time steps. The fittest strings in the population were selected to survive and were randomly paired to produce offspring by crossover, with each bit in the offspring subject to a small probability of mutation. This process was iterated for 100 "generations"—a "run"—with fitnesses estimated from a new set of ICs at each generation. Three hundred runs were performed starting with different random-number seeds. Details and justification of the experimental procedure have been given in ref. 9.

As reported previously (9, 12), the GA evolution proceeded through a succession of computationally distinct epochs. On most runs the end result was one of two computational strategies: settle to a fixed point of all 0s (1s), unless there is a sufficiently large block of adjacent or almost adjacent 1s (0s) in the IC; if so, expand that block. These strategies rely on the presence or absence of blocks as predictors of ρ_0. They do not count as sophisticated examples of emergent computation in CAs: all of the computation is done locally in identifying and then expanding a sufficiently large ($\approx 2r + 1$) block. After each run we computed a measure of the quality of the best rules in the final generation: the "unbiased performance" $\mathcal{P}_{149,10^4}(\phi)$, which is the fraction of correct classifications performed by rule ϕ within approximately $2N$ time steps with $N = 149$ over 10^4 ICs randomly chosen from an unbiased distribution over ρ. The unbiased distribution meant that most ICs had $\rho_0 \approx 1/2$. These are the most difficult cases, and thus $\mathcal{P}_{N,10^4}(\phi)$ gives a lower boundary on other measures of a rule's performance. The highest measured $\mathcal{P}_{149,10^4}(\phi)$ for block-expanding rules was 0.685 ± 0.004. Performance decreased dramatically for larger N, since the size of the block to expand and the velocity of expansion was tuned by the GA for $N = 149$ (see ref. 9). In general, any rule that relies on spatially local properties will not scale well with lattice size on the $\rho_c = 1/2$ task. This is shown in Table 1 for a typical block-expanding rule ϕ_{exp} discovered by the GA.

A major impediment for the GA was an early breaking of symmetries in the $\rho_c = 1/2$ task for short-term gain in fitness by specialization for high or low density (9, 12). This and other impediments seemed to indicate that this evolutionary system was incapable of discovering higher performance CAs. However, we subsequently discovered that in 7 of 300 runs, the GA evolved significantly more sophisticated methods of emergent computation. Again, the evolution proceeded via a series of epochs connected by distinct computational innovations. (A detailed analysis of the evolutionary history will be presented elsewhere.) $\mathcal{P}_{N,10^4}(\phi)$ values for three values of N are shown in Table 1 for the best rules (ϕ_{11102}, ϕ_{17083}, ϕ_{100}) in three of these runs. The higher $\mathcal{P}_{N,10^4}(\phi)$ values and the improved scaling with increasing N indicates a new level of computational sophistication above that of the block-expanding rules. Also given for comparison are two human-designed CAs: ϕ_{maj} computes the local majority of 1s in the neighborhood and, since it maps almost all configurations to small stationary blocks of 1s and 0s, has $\mathcal{P}_{N,10^4}(\phi) = 0.000$ for all N; ϕ_{GKL}, one of the best performing rules known, has the highest performance listed, though it was constructed not for the $\rho_c = 1/2$ task but for a study of ergodicity and reliable computation in CAs (13). Space-time diagrams illustrating the behavior of ϕ_{17083} and ϕ_{100} are given in Fig. 1. The space–time behavior of ϕ_{100} is remarkably similar to that of ϕ_{GKL} (see ref. 12). Its lower performance arises from factors such as asymmetries in the rule table.

How are we to understand the emergent computation these more successful CAs are performing? In previous work (14–16), we developed automated methods for discovering computational structures embedded in space–time behavior. Like many spatially extended natural processes, cellular automata configurations often organize over time into spatial regions that are dynamically homogeneous. Typically, the discovery of the underlying regularities requires automated inference methods. Sometimes, though (e.g., Fig. 1), these regions are obvious to the eye as "domains": regions in which the same recurring "pattern" appears. To understand this phenomenon and to automate its discovery, the notion of "domain" was formalized (15) by adapting computation theory to CA dynamics. There, a domain's "pattern" is described by using the minimal deterministic finite automaton (DFA) (17) that accepts all and only those configurations that appear in the domain. Such domains

Table 1. Measured values of $\mathcal{P}_{N,10^4}(\phi)$ at various N for six different $r = 3$ rules, the middle four discovered during different runs of the GA

CA ($r = 3$)	Symbol	Rule table hexadecimal code	$\mathcal{P}_{149,10^4}$	$\mathcal{P}_{599,10^4}$	$\mathcal{P}_{999,10^4}$
Majority	ϕ_{maj}	00010117 01171777			
		01171777 177f7fff	0.000	0.000	0.000
GA-discovered					
Expand 1-blocks	ϕ_{exp}	05054083 05c90101			
		200b0efb 94c7cff7	0.652	0.515	0.503
Particle-based	ϕ_{11102}	10000224 41170231			
		155f57dd 734bffff	0.742	0.718	0.701
	ϕ_{17083}	03100100 1fa00013			
		331f9fff 5975ffff	0.755	0.696	0.670
	ϕ_{100}	05040587 05000f77			
		03775583 7bffb77f	0.769	0.725	0.714
GKL	ϕ_{GKL}	005f005f 005f005f			
		005fff5f 005fff5f	0.816	0.766	0.757

For $N = 149$, the standard deviation is 0.004; it is higher for larger N. ϕ_{exp} expands blocks of 1s; ϕ_{maj} computes the majority of 1s in the neighborhood; all of the other rules implement more sophisticated strategies involving particle interactions. To recover the 128-bit string giving the CA look-up table outputs, expand each hexadecimal digit to binary. The neighborhood outputs then are given in lexicographic order starting from neighborhood 0000000 at the leftmost bit in the 128-bit binary string. ϕ_{GKL} is a rule designed by Gacs–Kurdyumov–Levin (see ref. 13).

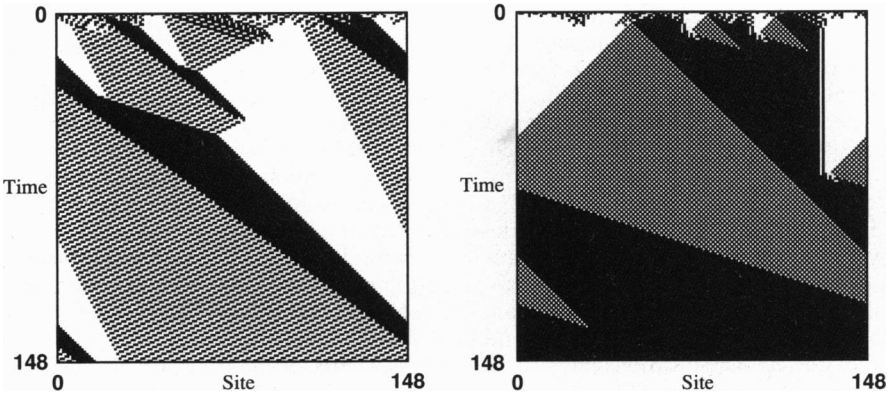

FIG. 1. Space–time diagrams showing the behavior of two CAs, discovered by the genetic algorithm on different runs, that employ embedded particles for the nonlocal computation required in density classification. Each space–time diagram plots lattice configuration iterates over a range of time steps, with 1s given as black cells and 0s as white cells; time increases down the page. Both start with the same initial configuration ($\rho_0 \approx 0.483$). (*Left*) ϕ_{17083} correctly classifies this low-ρ configuration by going to a fixed all-0s configuration by time 250 (not shown) after the gray region dies out. (*Right*) In contrast, CA ϕ_{100} misclassifies it by going to all 1s, despite its better average performance.

are called "regular," since their configurations are members of the regular language recognized by the DFA. More precisely, a regular domain Λ is a set of configurations that on an infinite lattice is temporally invariant, $\Lambda = \phi(\Lambda)$, and whose DFA has a single recurrent set of states that is strongly connected.

Regular domains play a key role in organizing both the dynamical behavior and the information-processing properties of CAs. Once a CA's regular domains have been detected (i.e., that level of structure has been understood), nonlinear transducers (filters) can be constructed to remove them, leaving just the deviations from those regularities. The resulting filtered space–time diagram reveals the propagation of domain "walls." If these walls remain spatially localized over time, then they are called "particles" (16). (We emphasize that such embedded particles are qualitatively different from those exhibited by CAs that have been hand-designed to perform computations (e.g., see refs. 18–21). Embedded particles are a primary mechanism for carrying information (or "signals") over long space–time distances. This information might indicate, for example, the result of some local processing that has occurred at an early time. Logical operations on the signals are performed when the particles interact. The collection of domains, domain walls, particles, and particle interactions for a CA represents the basic information-processing elements embedded in the CA's behavior—the CA's "intrinsic" computation.

CA ϕ_{17083} of Fig. 1 *Left* has three domains $\{\Lambda^0, \Lambda^1, \Lambda^2\}$, which are given in Table 2. There are five stable particles $\{\alpha, \gamma, \delta, \eta,$ and $\mu\}$ and one unstable "particle" $\{\beta\}$ defined in Table 2 as walls between two domains. Note that, given the CA rule code (Table 1), it can be proved that the domains are time-invariant sets and that the stable particles are spatially localized time-invariant sets for the corresponding CA (16). With this knowledge, the space–time diagram of Fig. 1 *Left* can be filtered to remove the domains. The result, shown in Fig. 2, reveals the particles and their interactions. Table 2 lists the six particle interactions that have been identified. The filtering analysis reveals a particle-based logic that emerges over time and supports the required computational processing—information storage and propagation over large space–time distances, logical operations, and so on—necessary for high fitness in approximating density classification. Roughly, ϕ_{17083} successively classifies "local" densities with a locality range that increases with time. In regions where there is some ambiguity, signals (in the form of particles) are propagated, indicating that the classification is to be made at a larger scale via particle interactions. Two examples of such interactions are shown in Fig. 2 *Left* and explained in the legend.

There are a number of constraints imposed by the "cellular" nature of CAs that the GA balances in its evolutionary search for high fitness. First, classification of local configurations with ambiguous density must be deferred to later times and larger spatial scales to provide a context in which information is available to disambiguate the local classification. Second, signals are required in the form of propagating particles, since local operations at later times have to be spatially local: decisions are made when particles come within an interaction range set by the CA radius. Third, the particle interactions must be built into the look-up table, which adds constraints that are nonlocal in the genomic representation and that must be compatible with domain stability and particle propagation. Fourth, the particles must be stable to preserve information over space–time. The result is a delicate balance that must be maintained by the GA in a CA look-up table that supports sophisticated particle-based information processing. Given these constraints, which are nonlocal and require specific output bit settings in the rule table string, it is striking that the

Table 2. The domains, particles, and particle interactions that support the emergent logic in the CA (ϕ_{17083}) shown in Fig. 1 *Left*

	Particle			Particle interaction (by type)		
Domain	Symbol	Velocity	Graphics	Annihilation	Decay	Reaction
$\Lambda^0 = 0^*$	$\alpha \approx \Lambda^1\Lambda^0$	1	α γ	$\delta + \gamma \rightarrow \emptyset$	$\beta \rightarrow \delta + \eta$	$\alpha + \delta \rightarrow \mu$
$\Lambda^1 = 1^*$	$\beta \approx \Lambda^0\Lambda^1$	0		$\mu + \eta \rightarrow \emptyset$		$\mu + \gamma \rightarrow \alpha$
$\Lambda^2 = (10001)^*$	$\gamma \approx \Lambda^2\Lambda^0$	−2				$\eta + \alpha \rightarrow \gamma$
	$\delta \approx \Lambda^0\Lambda^2$	½				
	$\eta \approx \Lambda^2\Lambda^1$	⅔	δ η μ			
	$\mu \approx \Lambda^1\Lambda^2$	3				

$(\omega)^*$ means any number of repetitions of string ω. The table includes only those structures that dominate the CA's spatiotemporal behavior. Very infrequently occurring structures, such as the checkerboard domain $\Lambda^3 = (10)^*$ and the four dislocations within Λ^2 are not listed because they do not contribute measurably to the CA's classification performance. Under "Particles," the graphic associated with each particle provides a key to Fig. 2. Note that the structure of each particle's graphic is determined by the nonlinear transducer. $\emptyset$ denotes spatial configurations without particles.

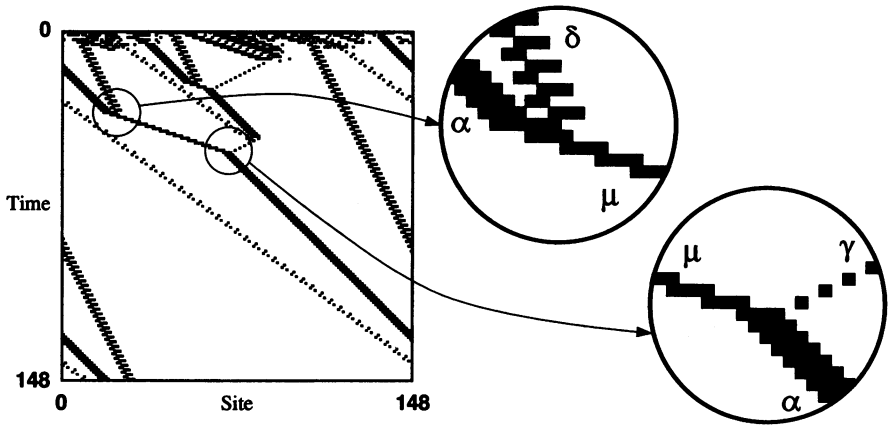

FIG. 2. Analysis of the emergent logic for density classification in CA ϕ_{17083} (Fig. 1 *Left*). This CA has three domains, six particles, and six particle interactions, as noted in Table 2. (*Left*) This figure gives the same space–time diagram as in Fig. 1 *Left* except that the domains have been filtered out by using an 18-state nonlinear transducer constructed as described in ref. 16. The resulting diagram reveals the particle interactions that support the long-range spatiotemporal correlation for density classification at the associated performance level (Table 1). (*Right Upper*) The particle interaction $\alpha + \delta \rightarrow \mu$ implements the "logic" of mapping a spatial configuration representing high, low, and then ambiguous densities to a high-density signal μ. (*Right Lower*) Similar detail is shown for the particle interaction $\mu + \gamma \rightarrow \alpha$ that maps a configuration representing high, ambiguous, and then low density to an ambiguous-density signal α.

GA evolved particle-based computation that performed nearly as well as the best-performing human-designed CA.

The particle-based computation analysis also indicates why the CAs discovered by the GA, as well as the human-designed CA, fail to achieve higher $\mathcal{P}_{N,10^4}(\phi)$. One reason, of course, is that the emergent logic can be incorrect. Even small errors in the particle velocities or interactions, for example, are compounded over time and lead to misclassifications. More importantly, at the very earliest iterations, before the CA behavior has condensed into configurations consisting only of domains and particles, local configurations larger than the neighborhood size can lead to incorrect positioning and selection of domains. The ensuing emergent logic operates on these errors and, even if it is correct, produces a misclassification. In this way, the analysis methods of refs. 15 and 16 allow us to explain how particular mechanisms in CAs lead to increased fitness and so survivability under the GA.

From the perspective of engineering applications, the particular GA used here was not an efficient automated designer of particle-based computation, since the rate of production of these CAs is low, though reliable. A primary impediment is the GA's breaking of symmetries in early generations for short-term fitness gain. This resulted in the populations' move to asymmetric, low-performance block-expanding CAs. Repairing the broken symmetries required an unlikely coordinated change in a large number of look-up table bits. We have proposed (9) a number of improvements to the GA, including the design of GA fitness functions and genomic representations that respect known task symmetries, but we also noted that symmetry-breaking may be a necessary part of some evolutionary pathways. On the subset of runs on which particle-based CAs were evolved, the GA was able to respect the symmetries necessary for higher performance and better scaling; this result and the success of our analysis of embedded computation are encouraging for the prospect of evolving more powerful particle-based computational systems for real-world tasks. Moreover, in work that will be reported elsewhere, the GA discovered perfectly performing CAs (on a high fraction of runs) that used particle-based computation on a different task: to rapidly achieve stable global synchronization between local processors.

The main result reported here is a simplified evolutionary process's discovery of methods for emergent global computation in a spatially distributed system consisting of locally interacting processors. Despite numerous phenomena that indicate nature has been confronted by analogous design tasks and solved them, to date only human-designed CAs have been used for performing such computations (see refs. 19–21). In contrast to the engineering approach of building particles and their interactions into CAs, a key tool in our analysis was the ability to detect structures embedded in CA spatiotemporal behavior that support emergent computation.

A simple, but general lesson was learned: when confronted with constraints, evolutionary processes need to innovate qualitatively new mechanisms that transcend those constraints. The locality of communication in CAs imposes a constraint on communication speed. The GA's innovation was to discover CAs that performed information processing over large space–time distances using particles and their interactions—a wholly new level of behavior that is distinct from the lower level of spatial configurations. In this way, our analysis of particle-based computation demonstrated how complex global coordination can emerge within a collection of simple individual actions. In a complementary fashion, our GA simulations demonstrated how an evolutionary process, by taking advantage of certain nonlinear pattern-forming propensities of CAs, can produce this new level of behavior through a succession of innovations that build up the delicate balance necessary for effective emergent computation.

We thank Rajarshi Das and Peter Hraber for many contributions to this project. This research was supported in part at the University of California at Berkeley by Air Force Office of Scientific Research Grant 91-0293 and Office of Naval Research Contract N00014-95-1-0524 and at the Santa Fe Institute by National Science Foundation Grant IRI-9320200, Department of Energy Grant DE-FG03-94ER25231, and the Adaptive Computation and External Faculty programs.

1. Pasteels, J. M. & Deneubourg, J. L., eds. (1987) *Experientia* **54,** Suppl.
2. Devreotes, P. (1989) *Science* **245,** 1054–1058.
3. Rumelhart, D. E., Hinton, G. E. & McClelland, J. L. (1986) in *Parallel Distributed Processing*, eds. Rumelhart, D. E., McClelland, J. L. & the PDP Research Group (MIT Press, Cambridge), Vol. 1, pp. 45–76.
4. Fama, E. F. (1991) *J. Finance* **46,** 1575–1617.
5. Milgrom, P. & Roberts, J. (1992) *Economics, Organization, and Management*. (Prentice-Hall, Englewood Cliffs, NJ).
6. Forrest, S. (1990) *Physica D* **42,** 1–11.
7. Crutchfield, J. P. (1994) in *Complexity: Metaphors, Models, and Reality*, Santa Fe Institute Studies in the Sciences of Complexity,

eds. Cowan, G., Pines, D. & Melzner, D. (Addison–Wesley, Reading, MA), Vol. 19, pp. 479–497.
8. Wolfram, S. (1984) *Nature (London)* **311,** 419–424.
9. Mitchell, M., Crutchfield, J. P. & Hraber, P. T. (1994) *Physica D* **75,** 361–391.
10. Land, M. & Belew, R. K. (1995) *Phys. Rev. Lett.* **74,** 5148–5150.
11. Holland, J. H. (1992) *Adaptation in Natural and Artificial Systems* (MIT Press, Cambridge), 2nd Ed.
12. Mitchell, M., Hraber, P. T. & Crutchfield, J. P. (1993) *Compl. Syst.* **7,** 89–130.
13. Gacs, P. (1985) *Contemp. Math.* **41,** 125–134.
14. Crutchfield, J. P. & Young, K. (1989) *Phys. Rev. Lett.* **63,** 105–108.
15. Hanson, J. E. & Crutchfield, J. P. (1992) *J. Stat. Phys.* **66,** 1415–1462.
16. Crutchfield, J. P. & Hanson, J. E. (1993) *Physica D* **69,** 279–301.
17. Hopcroft, J. E. & Ullman, J. D. (1979) *Introduction to Automata Theory, Languages, and Computation* (Addison–Wesley, Reading, MA).
18. Lindgren, K. & Nordahl, M. G. (1990) *Compl. Syst.* **4,** 299–318.
19. von Neumann, J. (1966) *Theory of Self-Reproducing Automata* (Univ. of Illinois Press, Urbana).
20. Smith, A. R. (1972) *J. Comput. Syst. Sci.* **6,** 233–253.
21. Steiglitz, K., Kamal, I. & Watson, A. (1988) *IEEE Trans. Comput.* **37** (2), 138–145.