Developmental Structuring of Phenotypic Variation: A Case Study with a Cellular Automata Model of Ontogeny

Wim Hordijk[*]
Konrad Lorenz Institute for Evolution and Cognition Research
Klosterneuburg, Austria
`wim@WorldWideWanderings.net`

Lee Altenberg
University of Hawai‘i at Mānoa
Honolulu, Hawai‘i, USA 96822
`altenber@hawaii.edu`

[*] Corresponding author

Abstract

Developmental mechanisms not only produce an organismal phenotype, but they also structure the way genetic variation maps to phenotypic variation. Here we revisit a computational model for the evolution of ontogeny based on cellular automata, in which evolution regularly discovered two alternative mechanisms for achieving a selected phenotype, one showing high modularity, the other showing morphological integration. We measure a primary variational property of the systems, their distribution of fitness effects of mutation. We find that the modular ontogeny shows the evolution of mutational robustness and ontogenic simplification, while the integrated ontogeny does not. We discuss the wider use of this methodology on other computational models of development, as well as real organisms.

1 Introduction

The idea that phenotypic variation could ever be "unbiased" is a historical artifact coming from two main sources: First were the early characterizations of quantitative genetic variation for single traits or small numbers of traits. Such measurements are actually projections of the extremely high dimensionality of organismal properties into very few dimensions. When the additive genetic variance was measured for these low-dimensional projections, positive additive genetic variance was discovered to be ubiquitous, and this finding was confirmed by selective breeding experiments that showed populations were widely responsive to artificial selection, so these trait values could be changed. The second step was the extrapolation of these low-dimensional observations to the entire organism, resulting in the adoption of the classical infinitesimal model of quantitative genetics (Fisher, 1918; Barton et al., 2017) as a plausible model for the entire, high-dimensional organismal phenotype, in which genetic variation is essentially a gas that can fill any selective phenotypic bottle (enunciated by Hill and Kirkpatrick (2010) who summarize, "almost anything can be changed if it shows phenotypic variation" and "combinations of traits, even those unfavorably correlated, can be changed"). Once the premises of this "pan-variationism" are accepted, pan-selectionism is the natural conclusion — in other words, if variation is diffusing in every phenotypic direction, the only source of information that shapes an organism's phenotype is natural selection.

Breaking away from the pan-variationist paradigm has been a prolonged process. Rupert Riedl's analysis was pioneering in this regard (Riedl, 1975, 1977, 1978), pointing out that the very ability to infer phylogenies based on phenotypes requires that natural selection be limited in its ability to erase the phenotypes of the past. Riedl proposed, furthermore, that complex phenotypes which require multiple simultaneous trait changes would often be impossible to evolve without the advent of genes that happen to make the right pleiotropic combination of multiple trait changes, and that such genes produce a genotype-phenotype map "in which gene interactions have imitated the patterns of functional interactions in the phenome." (Riedl, 1977).

In the 1980s came the identification of "developmental constraints" as sources of information other than natural selection that shape evolution (Bonner, 1982). Subsequently, it was recognized that development also focuses phenotypic variation along certain dimensions, thus enhancing the likelihood of evolution taking these paths (Roth and Wake, 1985). Together these have been combined into the concept of “developmental bias” (Yampolsky and Stoltzfus, 2001).

This term ‘bias’, however, still makes reference to the pan-variationist frame (Fillmore, 1976; Lakoff, 2014) that poses “unbiased” phenotypic variation as somehow a viable concept, and even perhaps a most-parsimonious null hypothesis.

When viewed from the vantage point of developmental mechanisms, it is clear that the processes that produce the physical structures of organisms will also structure the phenotypic variation that results from genetic and environmental variation. The idea of “unbiased” phenotypic variation is, from this vantage point, as sensible as the idea of “unstructured development” — which of course, makes no sense at all.

The question we need to be asking, then, is not how development *biases* phenotypic variation, but rather, how development *structures* it. This is the framework that we adopt here. It is hypothesized in Altenberg (2005) (and more recently Runcie and Mukherjee (2013)) that organismal variation actually lies on very low-dimensional manifolds embedded within the high-dimensional trait space. Within this framework, a principal open problem in evolutionary theory is to understand in what ways the developmental structuring of phenotypic variation may itself be systematically shaped by the evolutionary process. That is the question pursued here.

1.1 Ontogeny as a Dynamical System

Ontogeny in many animals exhibits several properties of an autonomous dynamical system, in which the trajectory of the system is set by the initial conditions. In common animal development, the mother creates the initial conditions for the zygote (often some form of egg), and development occurs in isolation from environmental interaction, until the organism “hatches”. Development may be less autonomous in placental mammals because of continuing material inputs from the mother, but most of the dynamical interactions of development occur internally within the embryo and fetus itself.

It is a generic property of dynamical systems that the trajectories converge toward attractors. Attractors may be fixed points, cycles, or strange attractors. Whole subsets of different initial conditions will converge toward the same attractor, comprising its domain of attraction. Different domains of attraction will converge toward different attractors.

The dynamics of epigenesis and ontogeny result from the complex interactions of the genome, oocyte cytoplasmic structure, and multicellular interactions once embryogenesis has begun. This entire process we can refer to as the *generative properties* of the genotype-phenotype map. The problem of concern here — how changes in the genome map to changes in the phenotype — can be referred to as the *variational properties* of the genotype-phenotype map. This distinction, introduced in Altenberg (1995), has been found to be useful in a number of studies (c.f. Jernvall and Jung (2000); Salazar-Ciudad (2006); Reiss et al. (2008); de la Rosa (2014); Porto et al. (2016); Laubichler et al. (2018)). The way the phenotype is generated from the genotype during development is clearly what structures phenotypic variation, so the variational properties are derived from the generative.

There is yet no general theory as to how evolution may shape the variational structure of phenotypic variation. Many computational models have been explored to try to tease out the regularities which may be eventually guide us to a general theory (Tsuchiya et al., 2016; Dong and Liu, 2017; Scholes et al., 2017). Here we revisit one such computational model which has produced intriguing results: the cellular automaton model of development, EvCA.

The cellular automaton is, as the name suggests, an autonomous dynamical system. Its dynamics are specified by initial conditions of its multi-variable state, and rules which determine the changes of the state from one time to the next. Cellular automata (CAs) exhibit complex attractors and domains of attraction, and moreover, allow one to watch their development progress. Here, we will define a task that the CA must achieve to stay "alive", and model a population of evolving CAs whose determining rules are mutated and recombined and subject to selection.

The distribution of fitness effects of mutation (DFE). Traditionally, the quantity of interest in evolutionary simulations has been the fitness of the digital organisms. Here our focus is on the variational properties of these organisms, and we investigate a principal quantification of how genetic variation maps to phenotypic variation, the *distribution of fitness effects of mutation* (DFE), which is the probability distribution over the changes in fitness caused by mutation of a given genotype.

The distribution of fitness effects of mutation provides a quantitative way to characterize the widely used concepts of mutational robustness, deleterious mutation, and evolvability. It is the probability distribution of mutant offspring's fitnesses relative to the parent's fitness. The distribution can be divided into three regions: (1) the lower tail, comprising deleterious mutations, (2) the measure near 1 (the parent's relative fitness), comprising the neutral mutations, which quantifies mutational robustness, and (3) the upper tail of the distribution, where the advantageous mutations occur, which characterizes evolvability. This quantification of evolvability was introduced in Altenberg (1994, 1995). Its use as a statistic to guide variation operators in evolutionary algorithms goes back to Rechenberg (1973), who introduced the integrated upper tail as the "success probability".

It should be noted that there has been controversy and confusion in the literature about whether evolvability should be consider a group property (such as population or species or lineage), or instead an individual property (c.f. Brown (2013)). The distribution of fitness effects of mutation is clearly a property belonging to an individual genotype, but it is measurable only by observing the fitnesses of the individual genotype's offspring. So it is never something we can measure in the individual itself, being rather an abstract "dispositional" property (Wagner, 1981) or "propensity" of the genotype. Also it should be noted that the DFE can change with each genotype in a succession of mutations, so the upper tail of the fitness distribution is only an immediate measure of evolvability by mutation, and it is necessary to know what happens to the upper tail along a sequence of fitter mutations to know what the long-term evolvability is.

An additional variational property is modularity — the extent to which genetic variability exists to vary different phenotypic properties as independent dimensions of variation. Here, we show that the evolution of both robustness and modularity can be observed (and analyzed) in the computational model of evolving cellular automata (EvCA). In the EvCA project, a genetic algorithm was used to evolve cellular automata to perform non-trivial computational tasks, with the aim of answering the general question: "*How does evolution produce sophisticated emergent computation in systems composed of simple components limited to local interactions?*" (Hordijk, 2013).

Recently, this EvCA framework was used to argue for *complexity by subtraction*, i.e., the idea that complexity may start out high and subsequently be reduced by natural selection without loss of functionality (McShea and Hordijk, 2013). It turns out that the observed "simplification" during evolution in the EvCA model is related to both robustness and modularity, as we will show in the current paper.

First, we present a brief review of the EvCA model, and explain the basics of cellular automata and genetic algorithms. Next, we show that the evolution of robustness and modularity can be explicitly observed, and how it can be analyzed in this model. It is our hope that these results and this methodology could lead to suggestions for actual experiments and predictions on the evolution of evolvability in actual biological systems.

2 Methods: Evolving cellular automata with a genetic algorithm

Cellular automata (CAs) are discrete dynamical systems which are simple to define but whose dynamics capture many essential mathematical properties of complex systems. For this reason, we use them as a model of development. They consist of an array of cells, each of which exists in one of a usually small number of states. The states of the array change over time. The change of the array over time is our model of development. Their next state in time of a cell in the array is a function of (1) its own current state and (2) those of a limited number of its local neighbors. With these simple dynamics CAs are able to produce a wide variety of intricate patterns in their aggregate dynamical behavior. These patterns are often considered to be *emergent*, in that they are not evident in the simple local rules of the individual components, but arise at a global level as their states progress in time.

We model the evolution of a developmental system by subjecting the cellular automata to a genetic algorithm (GA), which is a stochastic search and optimization method that is based on evolution by natural selection, and is therefore also often used as an actual computational model of an evolutionary process. Combining these two methods (GA and CA) provides a useful and versatile computational framework to study the evolution of robustness and modularity, as we will show below.

2.1 Cellular automata

Cellular automata were originally introduced by John von Neumann (after a suggestion by his colleague Stanislaw Ulam) to study the logic of self-reproduction (von Neumann, 1966; Burks, 1970). They were later popularized with John Conway's "Game of Life" (Gardner, 1970). Because of their intricate and emergent dynamical behavior, cellular automata are often used as simple computer models to study pattern formation, self-organization, and emergence.

In its simplest form, a *cellular automaton* (CA) consists of a linear array (or *lattice*) of identical "cells", each of which can be in one of two states, say zero or one. At each time step (or iteration), all cells simultaneously update their state according to a fixed update rule, depending on their current local "neighborhood configuration" which consists of the cell itself, its n left neighbors, and its n right neighbors, so that $2n+1$ cells determine its next state. This update rule simply states for each possible neighborhood configuration what the new state of the center cell will be. A rule, then, is a binary-valued map $f : \{0,1\}^{2n+1} \to \{0,1\}$. Given two possible states and a three-cell neighborhood, there are $2^3 = 8$ possible neighborhood configurations. An example of such an update rule, represented as a *look-up table* (or LUT), is given below. The top row lists the eight possible local neighborhood configurations, while the bottom row lists the corresponding new state of the middle cell.

neighborhood	000	001	010	011	100	101	110	111
new state	0	1	0	0	1	0	0	0

A two-valued CA with $n = 1$, which yields a neighborhood size of 3, is known as an *elementary cellular automaton* (ECA) (Wolfram, 1983), and the particular update rule shown above is known as ECA 18 out of the $256 = 2^8$ possible rules, since there are 8 possible neighborhood configurations, and two choices for how the rule maps each configuration. Depending on which particular rule is used, the system as a whole (the lattice of cells), can show very different types of dynamical behaviors, from fixed point or simple periodic, to very complex or even (seemingly) random behavior.

Figure 1 shows the dynamical behavior of ECA 18 in a so-called *space-time diagram*, with white representing the state zero and black representing the state one. In this diagram, space is horizontal and time is vertical. The top row (100 cells wide) is the *initial configuration* (IC), which in this case was generated at random. Each next row is the CA configuration after applying the update rule to all cells simultaneously, for a total of 100 iterations. Note that *periodic boundary conditions* are used, i.e., the lattice is considered to be circular so that the left-most cell and the right-most cell are each others neighbors.

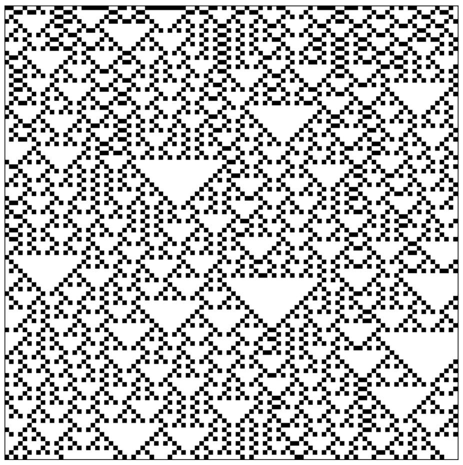


Figure 1: A space-time diagram of Elementary CA 18. Space is horizontal (100 cells) and time is vertical (100 iterations), starting from a random initial configuration (IC) in the top row. Periodic boundary conditions are used.

Of course there are many possible variations on the basic definition of cellular automata. For example, more than two states can be used, or a larger neighborhood size, or a higher-dimensional lattice. Each of these variations increases the size of the update rule, and thus also the total number of possible rules. Other variations such as asynchronous updates, non-uniform update rules, or randomness in the update rule can be included. The possibilities are endless, and so are the range and complexity of corresponding dynamical behaviors. For our purposes, a 2-state CA with neighborhood size 7 exhibits sufficiently rich behavior.

We do not attempt to identify actual biological interactions in development that resemble the CA rules, which would be an awkward fit at best. The relationship between this CA model of development, and actual organismal processes of development, is similar to that between artificial neural network models of cognition and the actual workings of the brain: it is on an dynamical structure level rather than a part-for-part analogy. The dynamical structure here is that cells communicate locally with other cells and interact according to certain rules, and we want to see if the rules can evolve so as to produce a desired collective behavior.

2.2 Genetic algorithms

Genetic algorithms are an extension of Darwinian evolution to structures beyond organisms and their DNA genotypes, to systems where the genotype may be any data structure, and the phenotype is produced by a computation performed upon this data structure. In using cellular automata as a model evolutionary system, we are treating the CA update rule (lookup table, LUT) as the "genotype", and the actual dynamical behavior of the CA (as visualized in a space-time diagram) as the corresponding "phenotype", and more specifically, as a model of ontogeny.

Moreover, we are using the computational capability of the CA as a model of how the organism can achieve a given phenotype despite randomness in its initial conditions and environment. In this way, the CA models the evolution of canalization, which is the stabilization of a developmental trajectory in the presence of environmental noise (Waddington, 1942). As our model phenotype we are building upon prior work in which the CA was evolved to produce a globally synchronized oscillator (Das et al., 1995; Hordijk et al., 1996, 1998).

As in natural evolution (at least to a first approximation), the random changes (mutation and recombination) happen at the level of the genotype, but fitness is determined at the level of the phenotype (the global CA dynamics). What makes this computational model of interest is that its genotype-phenotype map is analogous to the map from DNA genotype to organismal phenotype, in that it goes from the *specification* of a dynamical system to the *behavior* of the dynamical system.

A *genetic algorithm* (GA) (Holland, 1975; Goldberg, 1989; Mitchell, 1996) is a stochastic search method that tries to evolve better and better solutions to a given optimization problem by emulating Darwinian evolution and Mendelian genetics. The idea is to maintain a population of candidate solutions, suitably represented in a mathematical way, and create subsequent generations by applying selection, recombination, and mutation on the individuals (or their representation) in the current population, thus mimicking natural evolution. In each generation, the individuals (candidate solutions) in the population are applied to the given problem, and assigned a *fitness value* which indicates how well they solve the given problem. Based on these fitness values, they are then selected to "mate" and create offspring to make up the next generation.

This simulated evolutionary process generally leads to more and more fit (i.e., better and better) candidate solutions to the given problem. GAs have been used widely and successfully to find good approximate solutions to problems for which there is no analytical or efficient algorithmic way to find the best possible solution (including, to cite a highly publicized example, the recent production of the first image of a black hole by The Event Horizon Telescope Collaboration (2019)).

2.3 Evolving cellular automata with a genetic algorithm

Imagine the following computational task for a cellular automaton, known as *global synchronization*: from any initial configuration, the CA has to settle down to a synchronized temporal oscillation between an all-white configuration and an all-black configuration. This task was first described and studied in Das et al. (1995). Note that this is a non-trivial task for a CA, as it requires global information processing, even though each individual cell can only communicate locally (with its direct neighbors). It is inspired by the so-called *firing squad problem* (Moore, 1964), which is to get the cells in a CA to a synchronized state, and by natural phenomena such as the synchronous oscillations observed in fire flies, bacterial colonies, and the brain.

Here we utilize a cellular automaton with a local neighborhood of size 7, consisting of a cell itself and its nearest *three* neighbors on either side (called a *radius* of three). This gives rise to an update rule (LUT) with $2^7 = 128$ entries (possible neighborhood configurations), which means there are $2^{128} \approx 3.4 \times 10^{38}$ possible 2-state, radius-3 CA update rules, the number of possible "genotypes", a set equal in size to all possible 64 base long nucleotide sequences.

We apply an update rule to a lattice of 100 cells. The states of these 100 cells correspond to the state of a part of an organism as it develops in time. Thus there are $2^{100} \approx 1.27 \times 10^{30}$ possible "organismal" states. The question is then: Is there such a CA that can perform the global synchronization task, and if so, how do we obtain it?

Das et al. (1995) used a genetic algorithm to search through the astronomically large space of possible CA rules, resulting in CAs that can perform the global synchronization task perfectly. The population in this GA consisted of (initially random) CA update rules, represented by bit strings. The fitness value of each such a CA was then determined by iterating the corresponding update rule on a set of 100 random initial configurations (ICs), and simply counting on how many of these ICs it correctly settles down to the required globally synchronized oscillations within a maximum number of iterations. Figure 2 illustrates this population-based evolution and fitness calculation process graphically.

In a later study by McShea and Hordijk (2013) these experiments were repeated, and 100 GA runs on the global synchronization task were performed using the same parameter settings as in the original experiments (Das et al., 1995). This also resulted in several perfect rules. Here, we revisit this computational evolutionary system to investigate the evolution of variational properties of the genotypes (Altenberg, 1995), including robustness, canalization, modularity, and evolvability.

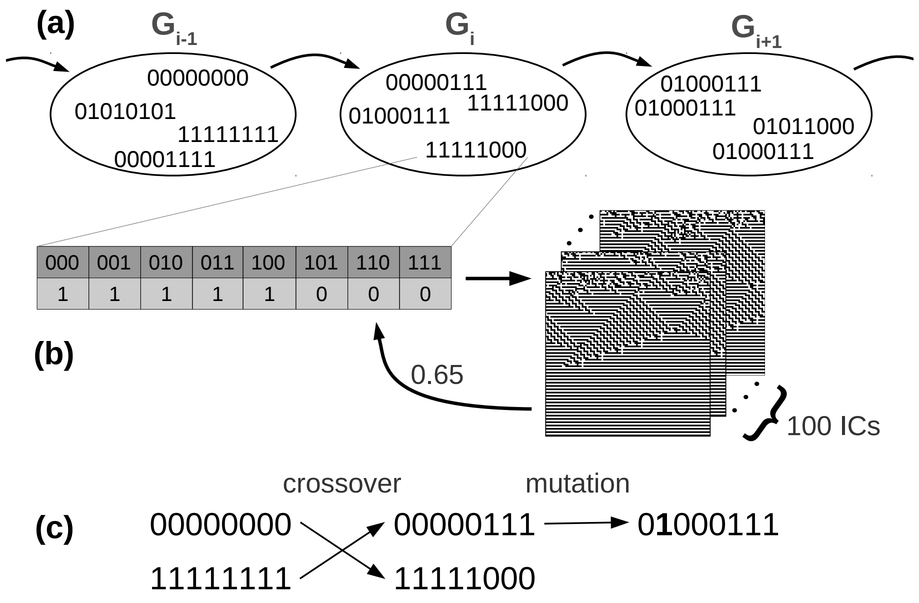


Figure 2: Evolving cellular automata with a genetic algorithm. (a) The GA population consists of bit strings, and changes from generation to generation due to selection, recombination, and mutation. (b) Each individual in the population represents a CA look-up table, which it iterated on a set of random initial conditions (ICs) to determine its fitness value on the given task. This fitness is calculated as the fraction of 100 random ICs on which it shows the correct behavior (e.g. 0.65). (c) Offspring individuals are created by applying recombination (crossover) and mutation on selected individuals (proportional to their performance measure) from the current population.

3 Results: The Developmental Structuring of Phenotypic Variation

The repeated experiment of 100 runs of the GA on the global synchronization task of McShea and Hordijk (2013) mainly resulted in three different "classes" of evolved CA rules:

1. Particle rules

2. Condense rules

3. Mixed (1 and 2)

Figure 3(a) shows an example of an evolved "particle rule". As was already described and analyzed in Das et al. (1995), such a CA rule uses emergent patterns in its dynamical behavior to resolve the conflict between locally oscillating regions that are out of phase with each other. In particular, there are the "regular domains", i.e., the regions with local oscillations between black and white, and the regions with alternating black and white L-shapes. Furthermore, "particles" (the boundaries between the regular domains) travel through the lattice at particular speeds and regularly collide, either annihilating each other or creating other particles, until eventually all out-of-phase conflicts are resolved and only one globally synchronized oscillating regular domain is left. More detailed analyses of such "particle strategies" in evolved cellular automata were presented in Hordijk et al. (1996, 1998).

Figure 3(b) shows an example of an evolved "condense rule". Such a CA rule does not make use of explicit particles, but somehow "condenses" into a globally synchronized state out of what seems to be almost random behavior. The third category consists of evolved CA rules that use a mix of these two behaviors (i.e., "particle" and "condense" combined). Out of the 100 GA runs, 28 runs resulted in particle rules (class 1), 36 in condense rules (class 2), and another 36 in mixed rules (class 3).

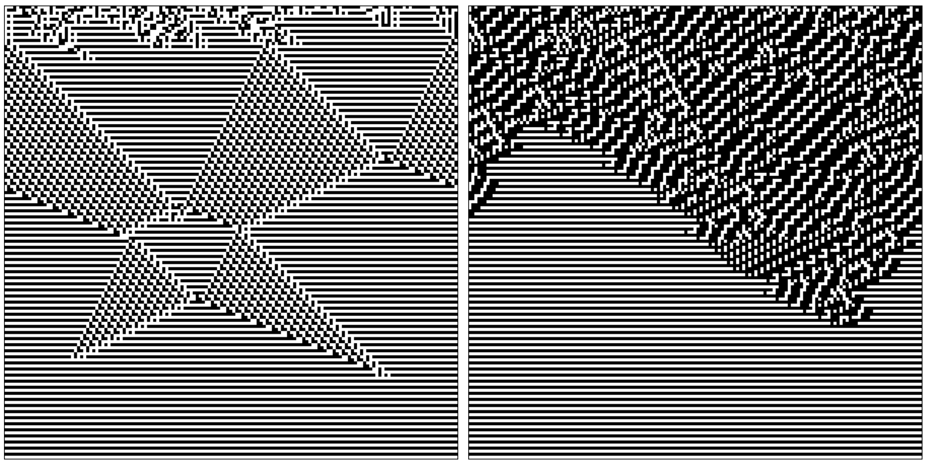


Figure 3: Space-time diagrams of (a) an evolved particle rule and (b) an evolved condense rule.

For all of these strategies, the CA has to take the random initial 100-cell array, which contains an average of around 70±11 out of the 128 possible 7-bit neighborhood patterns, and reduce the number of patterns in the array until only one is called in a give time increment, either 0000000 or 1111111. Particle rules are able to quickly reduce the number of patterns in the array to where they are all 0000000 and 1111111 and a 0/1 mixture representing the "boundaries" between the 0000000 and 1111111s domains. Condense rules take longer to reduce the number of patterns.

In McShea and Hordijk (2013) it was shown that in the case of the evolved particle rule shown in Figure 3(a), the particles (domain boundaries) actually simplified over time (i.e. during the GA evolution), both in number and in structure, i.e., in overall complexity. As was argued in McShea and Hordijk (2013), there is selective pressure for this to happen, given that less complex ("simpler") particles tend to make fewer mistakes in reaching the required final state. They thus exhibit greater canalization in Waddington's (1942) sense, in that the simplified domain boundaries reduce the fraction of initial conditions that go to the wrong attractor.

We now show how this observed reduction in particle complexity is directly related to the evolution of robustness and modularity.

3.1 Mutational Robustness and Developmental Architecture

For the investigation of how the variational properties of the CA evolve over time, we examine the distribution of fitness effects of mutation on the CA lookup table.

Recall that a CA update rule is represented by the *look-up table* (LUT), which lists all the possible local neighborhood configurations, together with the corresponding cell states at the next time step. For the 2-state, radius-3 (7-cell wide) neighborhood CAs used in the evolving cellular automata model, there are 128 entries in the LUT, each having either a 0 or a 1 to indicate the new cell state. For a LUT to solve the synchronization problem, the neighborhood 0000000 must output 1, and the neighborhood 1111111 must output 0.

Using the standard lexicographical ordering of the possible local neighborhood configurations (i.e., 0000000, 0000001, 0000010, and so on until 1111111), a CA update rule (or LUT) can thus be conveniently represented by a bit string of length 128 (i.e., the bit string made up of the bits in the bottom row of the LUT). This 128 bit string is the "genotype".

To show the evolution of robustness in cellular automata, first the best individual (CA rule) in each generation of a given GA run was taken (the GA was run for 100 generations in all experiments). For each of these "best-of-generation" CA rules, four statistics were then calculated:

1. Fraction of correct ICs (fitness).
2. The fitnesses of all 128 possible mutants of this best-of-generation rule.
3. Fraction of deleterious mutants.
4. Fraction of LUT entries used by the rule (a generative property of the CA).

Statistics 1, 3, and 4 range in value between 0 and 1.

Fraction of correct ICs: A CA's fitness value is simply the fraction of ICs on which it correctly reaches the globally synchronized oscillations within the maximum number of iterations. During the evolution of the CAs by the genetic algorithm, for the sake of computational time only 100 random initial configurations were sampled in determining fitness. For the statistic 1 in the list above, a more accurate estimation of the fitness of each CA rule was recalculated on a set of 10,000 random initial configurations.

Fitnesses of all 128 possible mutants: To obtain the distribution of fitness effects of mutation, individuals with all possible mutations of a single bit are generated, producing 128 mutant individuals, where a one-bit mutant is the original bit string representing the CA lookup table, but with one of the 128 bits flipped ("mutated"). The fitness of all 128 one-bit mutants was calculated (also on 10,000 random ICs). These distributions are depicted as they change during an evolutionary run in Figure 6 (see additional explanation in text).

Fraction of deleterious mutants: To focus on the mutational robustness of the CA, for each CA it was then calculated what fraction of its 128 one-bit mutants is *deleterious*, i.e., has a lower fitness than the parent CA itself, the converse of robustness. Given that there is some variance in the fitness calculations depending on which particular set of random ICs is used, a one-bit mutant is considered to be deleterious if its fitness is less than the original CA's fitness minus 0.01 (roughly the equivalent of one standard deviation on 10,000 ICs, i.e., $1/\sqrt{10,000}$). This provides statistic 3 above.

Fraction of LUT entries used: Our principal purpose here is to show how genotypes with different *generative* properties are associated with different *variational* properties. Recall that the generative properties are the ways that the genotype is actually utilized to produce the phenotype. Here we examine a very coarse-grained generative property of the CA, which is the fraction of LUT entries actually used during the iteration of the CA. An entry of the lookup table is called during execution of the CA when one of the 7-cell wide neighborhoods in the 100-cell array has a bit pattern corresponding to that entry. During the first time step of the CA with the random initial array, on average around 70 entries of the LUT are called (as mentioned earlier), for each of the 7-cell wide neighborhoods in the 100-cell array. When synchronization has finally been achieved, only two entries of the LUT are called, one for 0000000 and one for 1111111.

To compute the fraction of LUT entries called during iteration, a given CA is iterated for the allotted time on 100 random initial configurations, and each time a particular LUT entry is consulted (i.e., a cell finds itself in that particular local neighborhood configuration), that LUT entry's count is increased by one. In the random initial state, all LUT entries have an equal chance, 1/128, of being called. Under canalizing CA dynamics, each iteration tends to generate fewer and fewer neighborhood patterns. We avoid the initial period by skipping the first 10 iterations when counting the calls to the LUT entries. The counts over the remaining iterations are then averaged over all 100 initial configurations, and we course-grain the statistic by considering a LUT entry to be "used" if this average count is larger than 10. This provides statistic 4 above.

Figure 4(a) shows statistics 1, 3, and 4, averaged over the 28 GA runs that resulted in an evolved particle rule (class 1). The fitness of the best individual in each generation (i.e., the fraction of correct ICs; black line) increases rapidly in the first few generations, starts leveling off close to generation 20, and reaches a plateau by around generation 30.

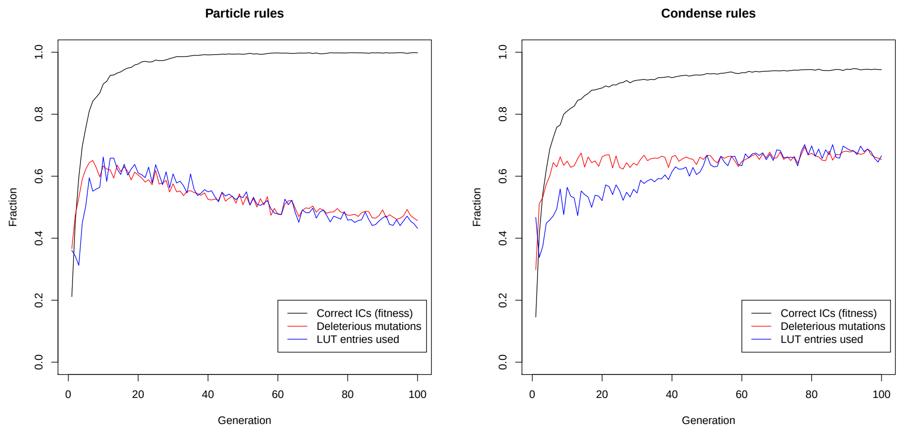


Figure 4: The fraction of correct ICs (fitness; black line), deleterious mutants (red line), and LUT entries used (blue line), for the best CA rule in each generation, averaged over the GA runs that resulted in (a) particle rules and (b) condense rules.

However, even though the fitness reaches a plateau, evolution at the genotype and developmental levels continues. As the red line in Figure 4 shows, the fraction of deleterious mutants initially increases (which is to be expected as genotypes get fitter), but once the fitness increase starts to level off (around generation 20), the fraction of deleterious mutants actually *decreases* again. And similarly for the fraction of LUT entries used (blue line).

This clearly shows the evolution of mutational robustness: once fitness has reached a plateau, there is selective pressure for CA rules to become more robust (i.e., have fewer deleterious mutants). This is achieved by reducing the number of LUT entries used to generate the required dynamical behavior (i.e., particles) to solve the global synchronization task, so that there is a lower chance of a mutated LUT entry ever being called by the CA.

The behavior here should be contrasted with that for simple genotype-phenotype maps such as Fisher's geometric model (Fisher, 1930), or Kingman's House-of-Cards mutation model (Kingman, 1977, 1978), in which the evolution of mutational robustness is impossible (Altenberg, 2015). Fisher's geometric model supposes that the phenotype is a point in an n-dimensional space, and its fitness increases the closer it is to a global optimum phenotype. Mutation samples the space in a ball around the parental phenotype. The closer the phenotype is to the optimum, the bigger is the fraction of that mutation-ball that takes the phenotype away from the optimum. Kingman's House-of-Cards mutation model supposes that the distribution of fitnesses from mutation is the same for all genotypes. Thus, the fitter the parent, the smaller the fraction of its mutant offspring that are still fitter than it. In both models, as evolution progresses, the fraction of deleterious mutations increases monotonically, while the fraction of advantageous mutations decreases monotonically.

The observed reduction in the number of LUT entries used during the "development" of the CA is obviously also directly related to the observed reduction in particle complexity (at the level of the phenotype) as shown in McShea and Hordijk (2013): simpler particles require fewer bits in the LUT.

Additional support for this claim is given in Figure 4(b), which shows the same three statistics averaged over the 36 GA runs that resulted in an evolved condense rule (class 2). In this case there is no evolution of robustness: both the fraction of deleterious mutants (red line) and the fraction of LUT entries used (blue line) level off in their ascent, but never decrease, not even after the fitness has reached a plateau.

We attribute this behavior to the nature of the "condense" strategy of synchronization. Synchronization is not achieved by simple domain boundaries interactions as in the "particle" strategies. Rather, synchronization results from a complex interaction of myriad neighborhood structures that is not easily understandable. A large diversity of neighborhoods are required for this mechanism, hence no decrease in the number of LUT entry calls as the population evolves. Because numerous LUT entries are required, mutation of any of them is more likely to result in a loss of fitness, and this limits the evolution of mutational robustness. This provides a striking example of how the mechanisms of development (here, the LUT entries) strongly structure the phenotypic effects of genetic variation.

This brings up the question whether a difference between the particle rules and condense rules can also be seen in the generative properties of their genotypes. Figure 5 shows such a comparison. It plots the probability that a 7-bit neighborhood is mapped to a 1 as a function of the number of 1s it contains. The horizontal axis in this figure shows the number of 1s in a local neighborhood configuration, ranging from zero to seven (recall that the local neighborhoods consist of seven cells). The vertical axis shows the fraction of those neighborhoods that have a 1 as the "output bit" (i.e., new cell state), averaged over all such neighborhoods for all 28 particle rules (yellow/gray) or all 36 condense rules (black), respectively.

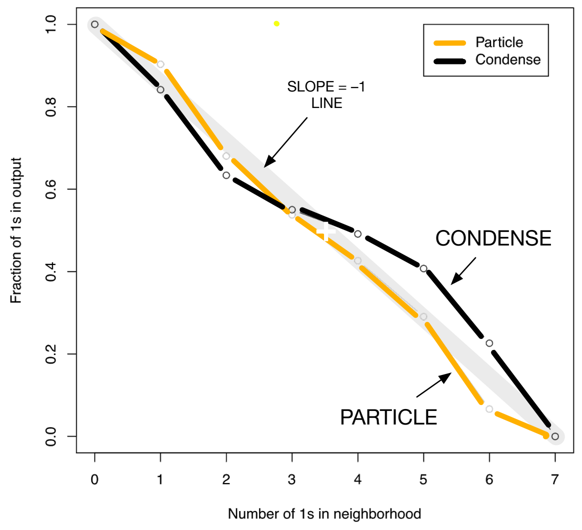


Figure 5: The fraction of 1s in the output bit for neighborhoods with different numbers of 1s, averaged over all particle rules (yellow/gray) or all condense rules (black), respectively.

To fulfill the synchronization goal, the neighborhood 1111111 must map to 0, and the neighborhood 0000000 must map to 1, so that the CA oscillates between all 0s and all 1s every iteration. This is seen at the endpoints of the curve, where the all-0s string maps to 1 with probability 1, and the all-1s string maps to 1 with probability 0 (i.e. it maps to 0).

The probability distributions for both Particle and Condense have average slope -1. This means that having an equal number of 1s and 0s in a neighborhood is an unstable state, and that an asymmetry in the number of 1s and 0s is magnified every iteration, on average, precisely the condition needed to get growing oscillations in composition.

We observe key differences, nevertheless, between the Particle and Condense curves. The slope of Condense is markedly less steep than the slope of Particle in the neighborhood configurations ranging from 2 to 5 ones, where 87% of the configurations lie ($\sum_{k=2}^{5}\binom{7}{k}/2^7$). This means that Particle rules more rapidly drive the CA away from a mixture of 1s and 0s toward the direction of all 1s or all 0s, where fewer distinct neighborhoods are possible. This is why there are fewer distinct calls made to the LUT in Particle than in Condense rules.

In other words, the particle rules "canalize" the outputs towards all 0s or all 1s faster during ontogeny than do the condense rules. This partly explains why particle rules tend to have a higher fitness, and why they are more robust.

3.2 Evolution of Mutational Robustness in Particle Rules

Mutational robustness (statistic 3) is a summary property of the distribution of fitness effects of mutation (DFE). However, our computational system enables us to observe the entire distribution (statistic 2) during evolution and see if it shows any systematic evolution.

Figure 6 shows the evolution of the DFE over 100 generations. It plots the distribution of fitness differences, $f_m - f_b$, between the best individual in each generation (f_b) and each of its 128 one-bit mutant offspring (f_m), for the same GA run from which the above analyzed CA rules were taken. Those differences are all plotted in sorted order.

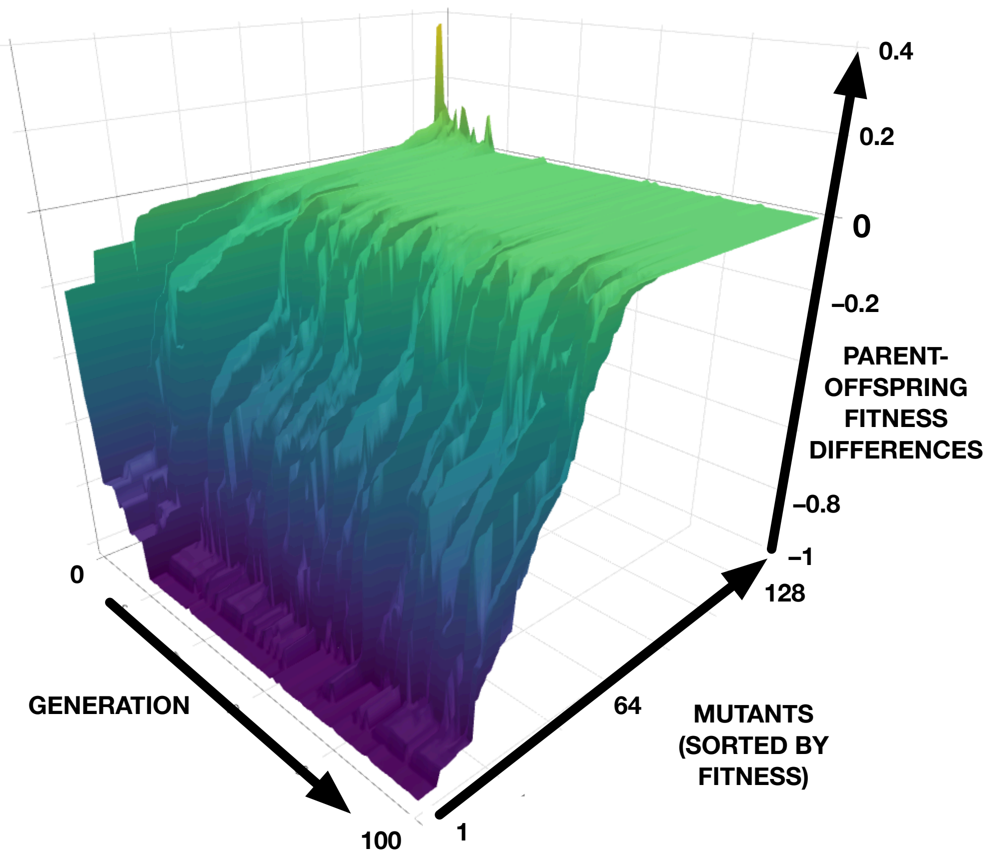


Figure 6: Evolution of the distribution of fitness effects of mutation during 100 generations of a GA run. Plotted are the fitness differences between the best CA rule in each generation and all of its 128 one-bit mutants. An interactive version that can be zoomed and rotated can be found on the following web page: `http://WorldWideWanderings.net/General/Mutants/index.html` . See the Supplementary Information for an explanation of why the flat areas correspond to modes of the distributions.

We first note the two large plateaus, one at the top (i.e., nearly neutral mutations around the parent CA's fitness), and one at the bottom. Plateaus in these sorted values are where many genotypes have nearly the same fitnesses, so the probability distribution near that value is high. That makes those values modes of the probability distribution. The way that a mode translates into a plateau is explained in the Supplementary Information.

The two plateaus in Figure 6, at top and bottom, reveal that the distribution of fitness effects of mutation is bimodal, with one mode near neutrality, and the other mode near lethality. It is notable that most every empirical study of DFEs finds similar bimodal distributions, with one mode near neutrality, and the other mode near lethality (Zeyl and DeVisser, 2001; Eyre-Walker and Keightley, 2007; Hogeweg, 2010; Hietpas et al., 2011; Bernet and Elena, 2015). This seems to be a general phenomenon, one that begs for a general theoretical explanation. (We should note that Fisher's geometric model can be made to produce a bimodal distribution with the right mean and variance values in the mutational distribution. Kingman's House-of-Cards model, on the other hand, can have a bimodal distribution only by construction).

The edge of the top plateau gives the proportion of deleterious mutants. It forms one of the data points that were averaged to produce the red curve in the right graph in Figure 4. The curve at generation 1 represent the mutant fitnesses of the best CA in the initial random population. The flatter slope here represents the fact that this initial best fitness is very low, and most of its mutant offspring have similarly low fitness. As soon as fitter mutants are generated, the mutational robustness decreases for about 10 generations, but then begins to increase, and reaches a steady distribution after about 20 generations. The average over many such distributions shows that the fraction of mutations that are deleterious continues to decline after generation 10 to the end of the runs at generation 100.

Figure 6 also exhibits an observed "notch" in the distribution, indicating a third, lower mode at a deleterious fitness of about -0.4. The cause and implications of this third mode are yet to be examined.

3.3 Further Out on the Adaptive Landscape

In the previous section we saw how the potential to evolve mutational robustness depends on the developmental strategies used to achieve a phenotype. The particle rules showed the continued evolution of mutational robustness even after reaching a fitness plateau, while the condense rules never evolved mutational robustness.

Next we will now take a closer look at the evolution of the adaptive landscape in the particle rules. We shall see that not only does the adaptive landscape change for the fittest evolved individuals, but it also changes for its *mutant offspring.*

We take a highly evolved particle rule and sample one of its deleterious mutant offspring, and compare that mutant's adaptive landscape to that of an individual from an earlier stage in evolution that has the same fitness as the mutant. From the GA run that produced the particle rule shown in Figure 3(a), we first took the best individual from the second generation, call it "Early", with only an intermediate fitness (0.5164). We then calculated the fitness values of all its one-bit mutants, and ordered those from smallest to largest value. The result is shown in Figure 7, with the fitness of the original CA rule indicated by the horizontal black line, and those of its 128 one-bit mutants by the black circles connected by a black line.

Next, we took the evolved particle rule shown in Figure 3(a), which was the best individual in the final generation of the GA run, and considered a particular one-bit mutant of it. This one-bit mutant, call it "Late", has a fitness of 0.5010, i.e., very close to that of the "Early" rule. Similarly, the fitness values of all its 128 one-bit mutants were then calculated and ordered from smallest to largest. The result is shown in Figure 7, with the fitness of the mutant CA rule indicated by the horizontal red line, and those of its 128 one-bit mutants by the red circles connected by a red line.

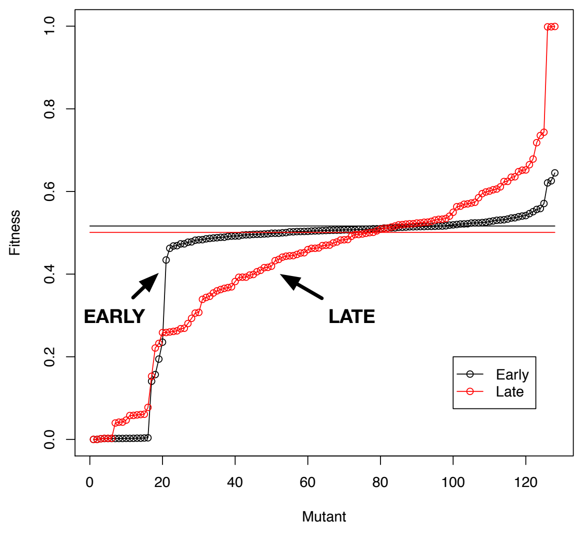


Figure 7: The CA fitness (horizontal line) and its mutant fitness values (circles) for an "Early" rule (black) and the mutant of a "Late" rule (red).

Several differences are immediately obvious. The distribution for "Early" has a very broad region of very little fitness change. Only about 17% of its mutants have a severe loss of fitness. But also only about 3% have a substantial fitness advantage. The "Early" genotype thus shows substantial mutational robustness, but a low level of evolvability (the upper tail of the distribution).

In contrast, the "Late" deleterious mutant has no broad plateau. Very few of its mutants are nearly neutral, but they show a wide spectrum of fitnesses, from zero up to 1.0, the fitness of its evolved parent. It therefore has very limited mutational robustness, but large evolvability.

The precise point at which each curve crosses its neutral mutation line is what determines the fraction of deleterious versus advantageous mutations. The "Early" black circles cross the horizontal black line around mutant 100, whereas the "Late" red circles already cross the horizontal red line around mutant 80, meaning that a greater proportion of the "Late" genotype's mutants are advantageous, indicating greater evolvability. But it is the "area under the curve" in the upper tail of the distribution that is dramatically larger in the "Late" genotype than in the "Early" genotype, meaning that there are many compensatory mutations in the "Late" genotype that can bring its fitness back to that of its evolved parent. There are actually *three* bits (the last three red circles) that can recover this "perfect" fitness in one mutational step.

In short, a mutant of a later CA with the same fitness as an earlier CA is clearly more more evolvable, but less robust to mutation, than that earlier CA. This shows that the adaptive landscape — as measured by the distribution of fitness effects of mutation — is not simply a function of an organism's fitness. Rather, the whole adaptive landscape in the vicinity of the highly evolved particle rule becomes shaped by evolution.

3.4 Evolution of Modularity in Particle Rules

The particle rules have evolved a developmental strategy of quickly canalizing random initial conditions into a small number of patterns. In the evolution of particle rules, we see that the patterns also evolve toward greater simplicity, which can be quantified here by the number of LUT entries that are called by the pattern in the space-time iteration. In contrast, the condense rules have evolved a developmental strategy in which a complex interactions of patterns ineluctably produce a synchronization of the cells. In the evolution of condense rules, we see in Figure 4(b) a continual *increase* in the complexity of their ontogeny, in that a greater and greater numbers of LUT entries become called as they evolve.

Here we take a closer look at the structure of the particle rules — their generative properties.

In McShea and Hordijk (2013) it was shown that there is a significant reduction in complexity of particles in CAs evolved for the global synchronization task, both in terms of number and in terms of structure. In particular, the CA shown in Figure 3(a) above, the best one in the final generation of a GA run, was compared with the best CA from generation 15 from the same GA run. Space-time diagrams for these two CAs are shown in Figure 8 below, adapted from McShea and Hordijk (2013, Fig. 9), where the regular domains are filtered out, clearly revealing the particles and their interactions. The different particle types are labeled with the letters **a** to **e**.

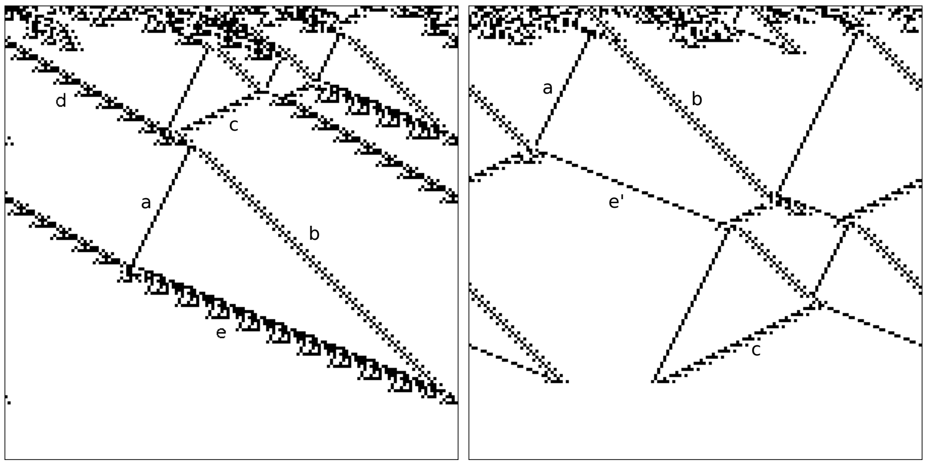


Figure 8: Space-time diagrams of two CAs, one from generation 15 (a) and one from generation 100 (b) from the analyzed GA run. The regular domains are filtered out to reveal the particles and their interactions more clearly.

Note that particles **a**, **b**, and **c** do not change during the evolution, i.e., they have the same structure in both CAs. However, particle **d** that exists in the best CA in generation 15 has disappeared in the best CA in generation 100. Furthermore, the structure of particle **e** in the earlier CA has simplified significantly in the later CA (labeled **e'** there). We now show how this reduction in complexity during the evolutionary process leads to modularity.

Each particle type is defined by, or "uses", a certain number of CA LUT entries. What is of specific interest here is how much overlap there is in this LUT entry usage between pairs of particles. The number of LUT entries used for each particle, and this overlap, are shown below. The columns contain the three unchanged particles **a**, **b**, and **c**, while the rows contain the particle that disappears (**d**) and the one that simplifies (**e** to **e'**). The numbers in parentheses in the last row and last column indicate the total number of LUT entries used by each particle, while the other numbers indicate the number of LUT entries each pair of particles has in common.

	a	b	c	#
d	5	7	8	(32)
e	7	1	6	(50)
e'	2	0	0	(10)
#	(8)	(12)	(13)	

As these numbers show, particle **d** not only uses a large number of LUT entries (32), but also has a significant overlap with the three particles **a**, **b**, and **c** (5, 7, and 8 LUT entries, respectively). However, this particle has disappeared in the later CA, thus greatly reducing the total amount of overlap between different particles. Furthermore, due to its reduction in complexity (from 50 to 10 LUT entries), the overlap of particle **e** with particles **a**, **b**, and **c** (7, 1, and 6, respectively) all but disappears (becoming 2, 0, and 0, respectively, for particle **e'**).

As a consequence, random mutations are much less likely to affect more than one particle at a time, giving rise to more robustness. Furthermore, should the CA need to adapt to changing circumstances (e.g., a new or modified task), we would expect that the remaining particles could now evolve largely independently, giving rise to increased evolvability. The particle rule CAs could therefore be considered to have discovered modularity as their means of evolving the synchronization function.

4 Conclusions and Discussion

We have employed a complex dynamical system, the cellular automaton, to explore how different evolved ontogenic strategies produce different properties of phenotypic variation. We would not call this "developmental bias" because there is no null hypothesis of what "unbiased variation" would look like for a dynamical system such as a cellular automaton. Rather, we seek to characterize the developmental structuring of the phenotypic variation by examining the ontogenic mechanisms that produce the phenotype. We have revisited the case of the evolving CA tasked with synchronizing all the cells of the CA into an oscillation for any random initial state, where it was discovered that two very different ontogenic mechanisms evolve, one the "particle" rule, and the other the "condense rule". The evolved "particle" rules are highly canalizing, and the initial random state of the CA array is quickly channeled into a very few patterns — domains and particle boundaries between them — where these boundary patterns collide and annihilate each other and leave their neighborhoods in a synchronized state. The evolved "condense" rules, on the other hand, find an ontogenic strategy in which a large number of patterns interact in a complex way that eventually leads to synchronization. These two strategies could be described as "parcellation" on the one hand — the fragmentation of the phenotype into modularly controlled features, and "integration" on the other — the production of a phenotype through the complex interaction of many contributing components (Wagner and Altenberg, 1996).

We examined a principal variational property of the phenotype, the distribution of fitness effects of mutation, and found that particle rules go through an evolutionary phase of integration followed by a phase of parcellation in which the mutational robustness increases, even after a perfect phenotype has evolved. Condense rules, on the other hand, continue an evolutionary trajectory of greater integration, in which more and more lookup table entries become invoked to achieve the synchronization task on which fitness is based. The condense rules do not show any evolution of mutational robustness, but in fact generate greater numbers of deleterious mutations as evolution and integration continue.

What we see in this case study is that different developmental strategies of producing a phenotype structure the phenotypic effects of genetic variation in different ways, and also facilitate (in the case of particle rules) or block (in the case of condense rules) the potential of evolving one fundamental variational property, the evolution of mutational robustness.

The value of this case study is that it points out new phenomena to investigate in other models of development, and potentially in model organisms. Cellular automata have been used to model diverse biological phenomena (Peak et al., 2004). This particular cellular automaton model has proven of interest to the cellular automata field in its own right, but like other useful model systems, it is one that allows for experimental manipulation and detailed measurement, with the hope that the phenomena it reveals have wider relevance.

The question arises as to whether the CA model explored here is a close enough analogy to any actual developmental processes to enable it to provide more specific biological insights than the general narrative we have discussed. The biological analogy of calls to the lookup table would be the expression of genes. The ideal application of this analogy would be to find an organismal phenotype that is generated with the expression of a small number of genes in one species, and a large number of genes in another species. Biological examples of such phenotypes undoubtedly exist, but we must leave it to developmental biologists to bring them to our attention. Certainly differences exist within single organisms between phenotypes that require the expression of large numbers of genes, and those that are generated with a small gene expression profile. Attempts to quantify the ubiquity of these two ends of the spectrum have come to different conclusions, from the finding of high modularity in Wagner and Zhang (2011), to the finding that almost every gene affects many characters (universal pleiotropy) and almost every character is affected by many genes, summarized as the *omnigenic model* of the genotype-phenotype map (Boyle et al., 2017). Without wading into the details of such studies, the behavior of the CA model studied here suggests, at least, that it may be worthwhile to investigate how the distribution of fitness effects of mutation may differ in traits exhibiting high polygeny versus those generated with low polygeny. In our investigation, only a single trait of the cellular automaton is under selection — the fraction of inputs that are correctly synchronized — so our results do not address questions of pleiotropy. However, there are many properties of the CA behavior that could also be placed under selection, so evolutionary CAs have the potential to serve as computational systems to investigate questions of how evolution may shape pleiotropy, and vice versa.

Here we have examined and performed artificial selection on only one phenotypic property of the CA, the proportion of initial conditions that the CA successfully develops into synchronized oscillations. Many other characteristics of the CA could also be examined; the complex sequence of states that the CA progresses through in its ontogeny allow us to dissect the phenotype into many other characters. We could examine, for example, the ontogenic trajectories of the fraction of states that are synchronized, or the distribution of calls to the 128 LUT entries. Each of these quantities is also probabilistic, displaying some distribution over the random initial conditions. A higher-dimensional analysis of how the developmental mechanisms of this CA model structure its production of phenotypic variation would be merited for any additional phenomena it may uncover.

An important direction to pursue is to expand the range of models beyond autonomous dynamical systems. In many cases, the phenotype consists of a map between environmental inputs and developmental, physiological, and behavioral responses. We would propose that perhaps the most inclusive formulation of the genotype-phenotype map — one that subsumes phenotypic plasticity and developmental noise — is to consider a map from genotype $\times$ environment to a probability distribution over phenotypes. This would certainly be helpful with plant development, in which plant growth responds to the immediate properties of the physical circumstances, where the specific positions of branches and leaves are stochastic variables. In this class of models, we are no longer dealing with the geometry of attractors and domains of attraction, but to the geometry of complex mappings between environment inputs and organismal responses.

Many computational models of development have been studied in the literature. We would advocate revisiting these models to analyze how the distribution of fitness effects of variation, including mutation, recombination, gene duplication, and other operators, may evolve differently in different systems. The collective experience that may emerge from such studies will hopefully provide a basis for a more general theory for how development evolves to structure phenotypic variation.

Acknowledgments

WH and LA thank the Konrad Lorenz Institute, Klosterneuburg, for financial support through its Senior Fellowships. WH thanks the Theoretical Biochemistry Group at the University of Vienna for providing CPU time on their cluster to perform the mutant analyses. LA thanks Marcus W. Feldman for support from the Stanford Center for Computational, Evolutionary and Human Genomics, and the Morrison Institute for Population and Resources Studies, Stanford University.

Conflict of interest

The authors declare no conflict of interest.

References

Altenberg, L. (1994). The evolution of evolvability in genetic programming. In Kinnear, K. E., editor, *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press, Cambridge, MA.

Altenberg, L. (1995). Genome growth and the evolution of the genotype-phenotype map. In Banzhaf, W. and Eeckman, F. H., editors, *Evolution and Biocomputation: Computational Models of Evolution*, volume 899 of *Lecture Notes in Computer Science*, pages 205–259. Springer-Verlag, Berlin.

Altenberg, L. (2005). Modularity in evolution: Some low-level questions. In Callebaut, W. and Rasskin-Gutman, D., editors, *Modularity: Understanding the Development and Evolution of Natural Complex Systems*, pages 99–128. MIT Press, Cambridge, MA.

Altenberg, L. (2015). Fundamental properties of the evolution of mutational robustness. *arXiv preprint arXiv:1508.07866*.

Barton, N., Etheridge, A., and Véber, A. (2017). The infinitesimal model: Definition, derivation, and implications. *Theoretical Population Biology*, 118:50–73.

Bernet, G. P. and Elena, S. F. (2015). Distribution of mutational fitness effects and of epistasis in the 5'untranslated region of a plant rna virus. *BMC Evolutionary Biology*, 15(1):274.

Bonner, J. T., editor (1982). *Evolution and Development: Report of the Dahlem Workshop on Evolution and Development, Berlin 1981, May 10-15*, volume 22 of *Life Science Research Reports*. Springer-Verlag, New York.

Boyle, E. A., Li, Y. I., and Pritchard, J. K. (2017). An expanded view of complex traits: From polygenic to omnigenic. *Cell*, 169(7):1177–1186.

Brown, R. L. (2013). What evolvability really is. *The British Journal for the Philosophy of Science*, 65(3):549–572.

Burks, A. W., editor (1970). *Essays on Cellular Automata*. University of Illinois Press.

Das, R., Crutchfield, J. P., Mitchell, M., and Hanson, J. E. (1995). Evolving globally synchronized cellular automata. In Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343. Morgan Kaufmann.

de la Rosa, L. N. (2014). On the possible, the conceivable, and the actual in evolutionary theory. *Biological Theory*, 9(2):221–228.

Dong, P. and Liu, Z. (2017). Shaping development by stochasticity and dynamics in gene regulation. *Open Biology*, 7(5):170030.

Eyre-Walker, A. and Keightley, P. D. (2007). The distribution of fitness effects of new mutations. *Nature Review Genetics*, 8:610–619.

Fillmore, C. J. (1976). Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32.

Fisher, R. A. (1918). The correlation between relatives on the supposition of Mendelian inheritance. *Transactions of the Royal Society of Edinburgh*, 52:399–433.

Fisher, R. A. (1930). *The Genetical Theory of Natural Selection.* Clarendon Press, Oxford.

Gardner, M. (1970). The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, 223(120):123.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Hietpas, R. T., Jensen, J. D., and Bolon, D. N. (2011). Experimental illumination of a fitness landscape. *Proceedings of the National Academy of Sciences U.S.A.*, 108(19):7896–7901.

Hill, W. G. and Kirkpatrick, M. (2010). What animal breeding has taught us about evolution. *Annual Review of Ecology, Evolution, and Systematics*, 41:1–19.

Hogeweg, P. (2010). Toward a theory of multilevel evolution: Long-term information integration shapes the mutational landscape and enhances evolvability. In Soyer, O. S., editor, *Advances in Experimental Medicine and Biology*, volume 751 of *Evolutionary Systems Biology*, pages 195–224.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press. Second edition: MIT Press, 1992.

Hordijk, W. (2013). The EvCA project: A brief history. *Complexity*, 18(5):15–19.

Hordijk, W., Crutchfield, J. P., and Mitchell, M. (1996). Embedded particle computation in evolved cellular automata. In Toffoli, T., Biafore, M., and Leão, J., editors, *Proceedings of the Conference on Physics and Computation*, pages 153–158. New England Complex Systems Institute.

Hordijk, W., Crutchfield, J. P., and Mitchell, M. (1998). Mechanisms of emergent computation in cellular automata. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature–V*, pages 613–622. Springer-Verlag.

Jernvall, J. and Jung, H.-S. (2000). Genotype, phenotype, and developmental biology of molar tooth characters. *American Journal of Physical Anthropology: The Official Publication of the American Association of Physical Anthropologists*, 113(S31):171–190.

Kingman, J. F. C. (1977). On the properties of bilinear models for the balance between genetic mutation and selection. *Mathematical Proceedings of the Cambridge Philosophical Society*, 81(03):443–453.

Kingman, J. F. C. (1978). A simple model for the balance between selection and mutation. *Journal of Applied Probability*, 15:1–12.

Lakoff, G. (2014). *The All New Don't Think of an Elephant!: Know Your Values and Frame the Debate.* Chelsea Green Publishing.

Laubichler, M. D., Prohaska, S. J., and Stadler, P. F. (2018). Toward a mechanistic explanation of phenotypic evolution: The need for a theory of theory integration. *Journal of Experimental Zoology*, 330:5–14.

McShea, D. W. and Hordijk, W. (2013). Complexity by subtraction. *Evolutionary Biology*, 40(4):504–520.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press.

Moore, E. F. (1964). The firing squad synchronization problem. In Moore, E. F., editor, *Sequential Machines, Selected Papers*, pages 213–214. Addison-Wesley.

Peak, D., West, J. D., Messinger, S. M., and Mott, K. A. (2004). Evidence for complex, collective dynamics and emergent, distributed computation in plants. *Proceedings of the National Academy of Sciences U.S.A.*, 101(4):918–922.

Porto, A., Schmelter, R., VandeBerg, J. L., Marroig, G., and Cheverud, J. M. (2016). Evolution of the genotype-to-phenotype map and the cost of pleiotropy in mammals. *Genetics*, 204(4):1601–1612.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart.

Reiss, J. O., Burke, A. C., Archer, C., De Renzi, M., Dopazo, H., Etxeberría, A., Gale, E. A., Hinchliffe, J. R., de la Rosa Garcia, L. N., Rose, C. S., Rasskin-Gutman, D., and Müller, G. B. (2008). Pere alberch: Originator of evodevo. *Biological Theory*, 3(4):351–356.

Riedl, R. J. (1975). *Die Ordnung des Lebendigen: Systembedingungen der Evolution*. Parey, Hamburg and Berlin.

Riedl, R. J. (1977). A systems-analytical approach to macroevolutionary phenomena. *Quarterly Review of Biology*, 52:351–370.

Riedl, R. J. (1978). *Order in Living Organisms: A Systems Analysis of Evolution*. John Wiley and Sons. A translation of Riedl (1975) by R.P.S. Jefferies, Chichester.

Roth, G. and Wake, D. B. (1985). Trends in the functional morphology and sensorimotor control of feeding behavior in salamanders: an example of the role of internal dynamics in evolution. *Acta Biotheoretica*, 34(2-4):175–191.

Runcie, D. E. and Mukherjee, S. (2013). Dissecting high-dimensional phenotypes with bayesian sparse factor analysis of genetic covariance matrices. *Genetics*, 194(3):753–767.

Salazar-Ciudad, I. (2006). Developmental constraints vs. variational properties: how pattern formation can help to understand evolution and development. *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, 306(2):107–125.

Scholes, C., DePace, A. H., and Sánchez, Á. (2017). Combinatorial gene regulation through kinetic control of the transcription cycle. *Cell Systems*, 4(1):97–108.

The Event Horizon Telescope Collaboration (2019). First M87 Event Horizon Telescope results. VI. The shadow and mass of the central black hole. *The Astrophysical Journal Letters*, 875:L6 (44pp).

Tsuchiya, M., Giuliani, A., Hashimoto, M., Erenpreisa, J., and Yoshikawa, K. (2016). Self-organizing global gene expression regulated through criticality: mechanism of the cell-fate change. *PloS ONE*, 11(12):e0167912.

von Neumann, J. (1966). *Theory of Self-Reproducing Automata.* University of Illinois Press. (edited and completed by A. W. Burks).

Waddington, C. H. (1942). Canalization of development and the inheritance of acquired characters. *Nature*, 150:563–565.

Wagner, G. P. (1981). Feedback selection and the evolution of modifiers. *Acta Biotheoretica*, 30:79–102.

Wagner, G. P. and Altenberg, L. (1996). Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976.

Wagner, G. P. and Zhang, J. (2011). The pleiotropic structure of the genotype–phenotype map: the evolvability of complex organisms. *Nature Reviews Genetics*, 12(3):204.

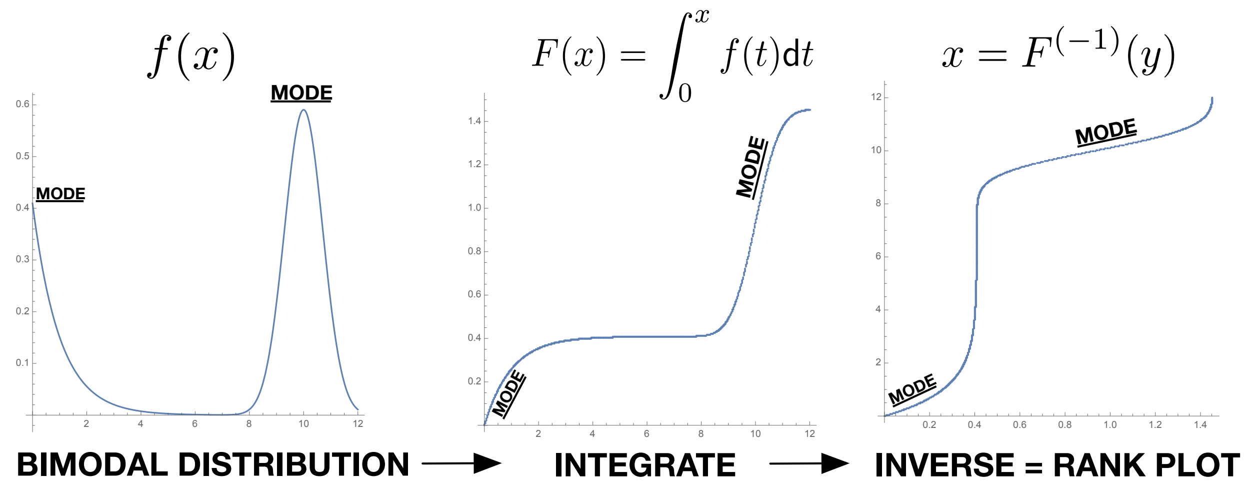
Wolfram, S. (1983). Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601.

Yampolsky, L. Y. and Stoltzfus, A. (2001). Bias in the introduction of variation as an orienting factor in evolution. *Evolution & Development*, 3(2):73–83.

Zeyl, C. and DeVisser, J. A. G. (2001). Estimates of the rate and distribution of fitness effects of spontaneous mutation in saccharomyces cerevisiae. *Genetics*, 157(1):53–61.

Supplementary information

How to interpret Fig. 6 in the main text. On the left is a bimodal probability distribution. This is integrated to give the cumulative distribution function (middle). This is then inverted to give the values as a function of rank (right). The modes on the left translate to flat areas on the right.


$f(x)$
$F(x) = \int_0^x f(t)\mathrm{d}t$
$x = F^{(-1)}(y)$
MODE
MODE
MODE
MODE
MODE
MODE
BIMODAL DISTRIBUTION
INTEGRATE
INVERSE = RANK PLOT