DOI: http://dx.doi.org/10.7551/978-0-262-31050-5-ch009

Evolution of Self-Replicating Cube Conglomerations in a Simulated 3D Environment

Paul Grouchy[1,2] and Hod Lipson[2]

[1]Institute for Aerospace Studies, University of Toronto, Toronto, ON, Canada M3H 5T6
[2]Mechanical & Aerospace Engineering, Cornell University, Ithaca, NY, USA 14850
paul.grouchy@utoronto.ca, hod.lipson@cornell.edu

Abstract

The evolution of self-replication in three dimensions is explored for the first time. A discrete three-dimensional world populated with physically-realizable "molecubes" is simulated. The cubes have randomly initialized controllers, can rotate about an axis, and can attach to one another to form conglomerations. Genetic material, which defines cube controllers, is exchanged stochastically between attached cubes and subject to random mutations. Self-replicating cube conglomerations emerge in this simulation across a wide range of densities and without the use of a fitness function, yielding insight into the evolution of self-replication in nature and furthering progress toward physically-realizable self-replicating machines.

Introduction

Researchers have been interested in artificial life simulations for as long as digital computers have existed. Early on, von Neumann invented cellular automata [Neumann (1966)], which are still an active area of research to this day. While the original cellular automata were programmed with the ability to self-replicate, more recent experiments have demonstrated the spontaneous emergence of replicators in such systems [Chou (1997)].

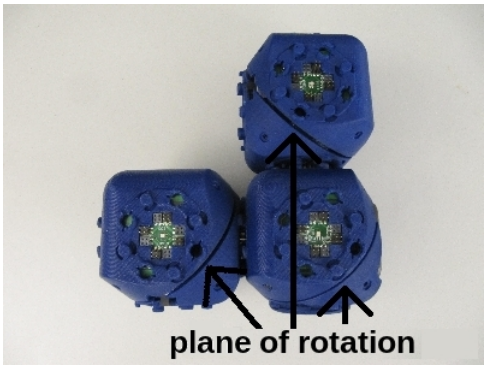


Figure 1: Three physical "molecubes". Note the plane of rotation.

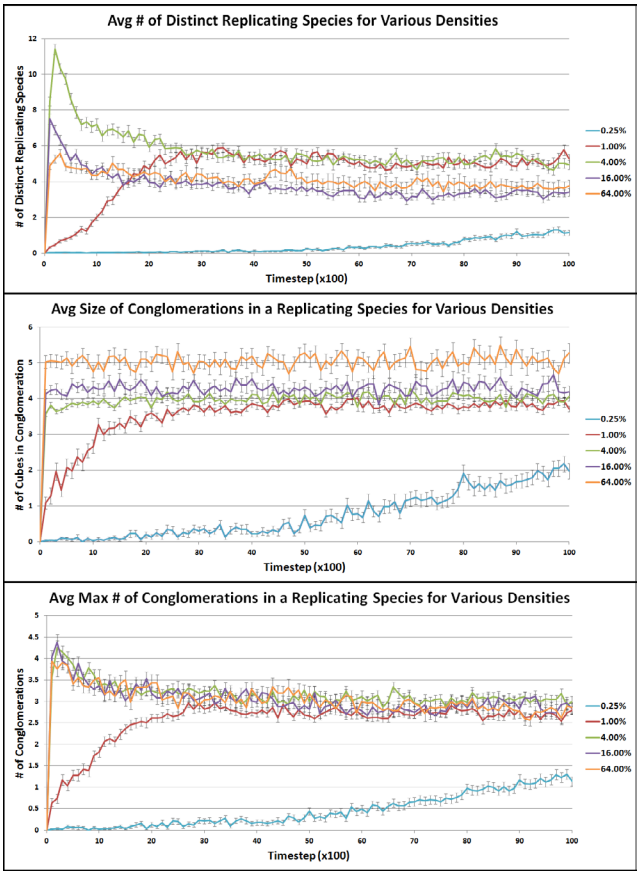


Figure 2: Experimental results for various densities. A replicating species is defined as a genome that occurs in two or more genetically homogeneous molecube conglomerations, where each conglomeration contains at least two cubes. Each result is an average over 100 randomly initialized runs and error bars show standard error.

In cellular automata simulations, every agent is identical (i.e. they all use the same ruleset). More complex artificial life paradigms such as Tierra [Ray (1992)] and Avida [Adami and Brown (1994)] simulate a diverse population of

© 2012 Massachusetts Institute of Technology

digital organisms that compete for computational resources, which can then be used for replication. Each agent in these simulations contains its own instruction set or "program" that can evolve over time. Organisms in Avida have the ability to self-replicate by running instructions to allocate memory for a child program and copy their instruction set into this memory. There is no explicit fitness function guiding evolution in these simulations, allowing for comparisons to self-replicating life on Earth. While further analogies can be drawn between these computational programs and real-world systems, it is difficult to imagine physical implementations of these artificial life-forms.

In an effort to narrow the gap between computational simulation and the physical world, a 2D simulation of non-uniform cellular automata that were physically realizable was designed and run in [Studer and Lipson (2005)]. The automata instruction sets existed in simulated "molecubes," which are cubes that can attach to one another using electromagnets and can rotate their halves around a fixed axis (see Figure 1). Physical versions of these cubes have previously been constructed and [Zykov et al. (2005)] demonstrated how a group of these molecubes could construct an identical second group using other molecubes. Preliminary results from the 2D simulations demonstrated, without the use of a fitness function, spontaneous emergence of self-replication. A group of simulated molecubes with identical rulesets (a "species") collected other molecubes in the 2D environment, transferred their rulesets, and then separated into two identical molecube groupings. A variety of self-replicating species often co-existed simultaneously, competing for molecube resources in the simulation.

The experiments presented in this paper bring ALife another step closer to realizable real-world systems by demonstrating the spontaneous emergence of self-replication in a population of physically realizable three-dimensional molecubes that exist in a simulated three-dimensional world. While this environment lacks several properties of the physical world, most notably gravity, this is the first time that the emergence of self-replication has been observed in three dimensions. Replicators emerged in simulations of varying densities, producing examples of agents that must move through the environment to accumulate cubes as well as replicators that were forced to remain largely stationary. This mirrors the independent rise of multicellularity in plants and animals [Bonner (1998)].

3D Physical Cube Automata

The simulated cubes in the following experiments were based on real "molecubes," presented in [Mytilinaios et al. (2004)]. Each of these physical cubes contains an actuator that allows it to rotate one of its pyramid-shaped halves in 120° increments and adjacent cubes can connect to one another using electromagnets. Adjacent cubes can also communicate over a digital channel. Figure 1 shows an example of these physical molecubes.

The computer simulations consisted of a population of simulated molecubes that exist in a three-dimensional NxNxN environment partitioned into a 3D grid. Each discrete grid location can either be vacant or occupied by a molecube. A single molecube cannot move from one discrete location to another, however a molecube can move other molecubes that are attached to it by rotating around its axis. One can then imagine various methods of locomotion whereby attached molecubes take turns rotating around their respective axis. Gravity is not incorporated into the simulation, therefore groups of molecubes can move in any direction. The simulated world wraps around, i.e. it is toroidal. If a molecube rotation creates a collision (i.e. two molecubes occupying the same 3D grid location), this move is reversed. To reduce the computational complexity of the system, collisions *during* a molecube rotation are ignored. Furthermore, a maximum of 15 molecubes could be attached together in a single group, and loops of attached molecubes were not allowed.

Each simulated molecube contains a controller that updates the cube's output set $\mathbf{y}$ based on its previous outputs and its current input values $\mathbf{x}$. See Table 1 for descriptions of the controller inputs and outputs. During a simulation, each molecube's controller is evaluated once per timestep. The order in which the controllers are evaluated is based on inter-molecube connections. Therefore while it is not random, it does vary over time.

The controllers used are 0D3v0 controllers [Grouchy and D'Eleuterio (2010)], where there is one evolvable ordinary differential equation per controller output y_n (see Equation 1).

$$d\mathbf{y}/dt = \mathbf{f}(\mathbf{x}, \mathbf{y}) \tag{1}$$

The functions f_n are represented as trees and can incorporate constants, inputs, outputs and a variety of mathematical operations (as in symbolic regression in Genetic Programming [Poli et al. (2008)]). For details on how the controllers are initialized, evaluated, and mutated, the reader is referred to [Grouchy and D'Eleuterio (2010)]. Crossover at the function level was not implemented for our experiments, however tree-level crossover that overwrites a randomly selected subtree with a randomly selected subtree from another controller was used. When at least one cube is attached to a cube selected for mutation, tree-level crossover is performed *instead of a mutation* with a probability of 0.5.

At each timestep, there is a probability μ that a random mutation will occur within a molecube's genome. Furthermore, if a molecube is attached to at least one other cube, there is a 50% chance that it will have its ODEs overwritten by an attached neighbours' ODEs. This can occur once per attached cube, per timestep. By stochastically deciding whether a cube's equations are to be overwritten by a

Parameter	Range	Description
$x_n, n \in [0,5]$	{0,1}	Incoming communication bit from molecube adjacent to side n (0 if no adjacent cube). Note that molecubes do not have to be attached to communicate.
$x_n, n \in [6,11]$	{0,0.5,1}	Adjacent/attached inputs. Set to 0 if no molecube adjacent to side $n-6$, 0.5 if molecube adjacent but not attached, 1 if molecube adjacent and attached.
$y_n, n \in [0,5]$	[0,1]	Outgoing communication bit to molecube adjacent to side n. If this output is greater than 0.5, a 1 is sent. Otherwise, a 0 is sent.
$y_n, n \in [6,11]$	[0,1]	Attach/detach output for side $n-6$. At each timestep, if a randomly generated value between 0 and 1 is less than the average of this output for two adjacent sides, their two respective molecubes are attached. Otherwise, they are detached.
y_{12}	[-1,1]	Molecube rotation output. If $-0.33 < y_{12} < 0.33$, the molecube does not rotate. The remainder of the output range is equally divided to represent the four possible rotations, two directions per half.

Table 1: Simulated molecube controller inputs $\mathbf{x}$ and outputs $\mathbf{y}$.

neighbour's, the inherent bias in the cube evaluation order is lessened.

Experiments

The goal of the experiments presented in this paper was to observe self-replicating cube "species" in a simulated three-dimensional environment. Here, a replicating species is defined as a genome that occurs in two or more genetically homogeneous molecube conglomerations, where a molecube conglomeration is defined as a grouping of two or more attached molecubes. Genetic distance was calculated as the sum of the tree edit distance between each output equation in a pair of genomes (tree edit distance was calculated using the Zhang-Shasha algorithm [Zhang and Shasha (1989)]). Self-replication is defined here as a series of actions whereby a genetically homogeneous molecube conglomeration accumulates molecubes from the environment and/or other conglomerations, overwrites their genomes with its own and then detaches at one or more points to produce two or more genetically homogeneous conglomerations that all contain the same genome. Self-replicating species are detected by searching the simulation for genomes that exist in two or more distinct, genetically homogeneous conglomerations. Note that the structures of the molecube conglomerations are ignored in this definition. This is owing to the fact that while genetically identical conglomerations were often observed, they were usually composed of a different number of molecubes, or the same number but arranged differently.

Experiments consisted of 1,000 randomly placed molecubes, each with a randomly generated genome. Experiments were performed with densities of 0.25%, 1%, 4%, 16% and 64% (note that in cases other than 1% density, the number of cubes had to be adjusted slightly to achieve the desired density). The mutation rate used was $\mu = 0.01$. An experiment would run for 10,000 timesteps, where a timestep consists of evaluating every molecube's controller, executing their outputs and stochastically performing mutations and equation overwrites. At periodic intervals, inter-conglomeration and between conglomeration genetic distances were calculated. If two or more genetically homogeneous conglomerations were found to contain the same genome, this species would be observed in a manually conducted test simulation. Test simulations would occur in smaller 3D grids (usually 9x9x9), populated by other conglomerations and/or single molecubes extracted from the same original simulation. The test simulation would last for 1,000 timesteps, and the results would be visualized using an RGB colour scheme to represent relative genetic distances. The goal of these test simulations was to observe self-replication. Furthermore, a variety of quantitative metrics based on genetic distance were used to analyze the simulations and to detect and observe the emergence of self-replicating species.

Results and Discussion

For the following results, data were collected at 100 timestep intervals. Figure 2 shows, for all experiments, the number of different self-replicating species detected at a given timestep (top), the average size of replicating conglomerations (middle), as well as the maximum number of conglomerations belonging to a single replicating species (bottom).

At low densities, replicators must be mobile to acquire new molecubes. At a density of 0.25%, very few replicating species arise, as there is little interaction between molecubes. Replicating species do appear on occasion, however they cannot acquire new molecubes fast enough to replicate further before succumbing to mutations. At 1% density, mobile conglomerations encounter new molecubes more frequently. Initially, a few small replicators appear. Over time, these initial replicators collect stationary molecubes, thus spreading genomes that promote conglomeration mobility. This also enables molecubes that were initialized without immediate neighbours to interact with other cubes. Thus, the molecubes in the system become more mobile, increas-

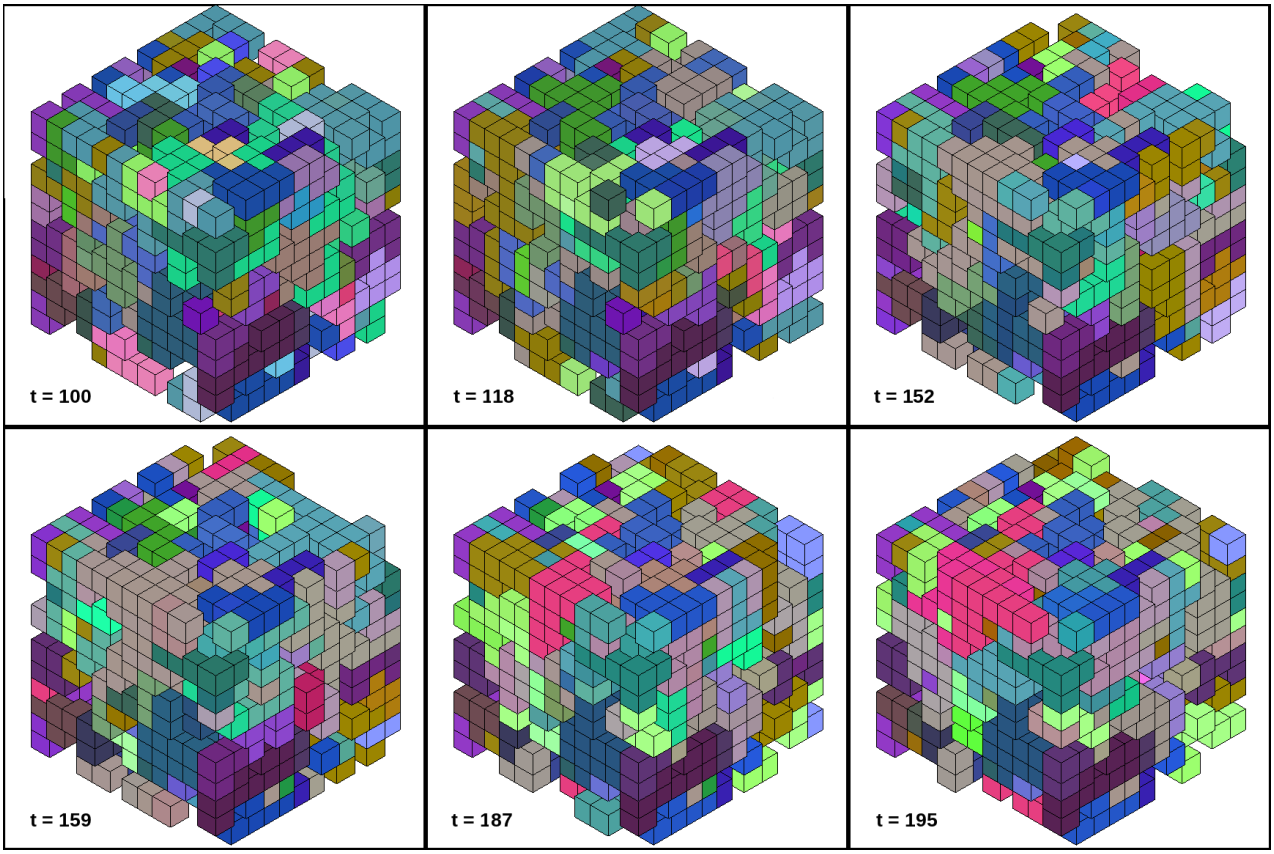


Figure 3: A collection of timesteps during a full simulation run at 64% density. Colours represent relative genetic distance. Despite their almost complete lack of mobility, several replicating species succeed in dominating large sections of the simulation world, albeit temporarily.

ing the number of molecube interactions, which in turn produces more replicating species and larger conglomerations.

At higher densities, molecubes are more likely to be initialized with adjacent neighbours, therefore a large number of replicating species appear within the first 100 timesteps. Interestingly, the number of distinct replicating species decreases as the simulation progresses, with the higher density simulations (16% and 64%) finishing with less distinct species on average than the lower density simulations (1% and 4%). This is most likely owing to the larger number of molecube interactions that will occur at higher densities, which in turn will lead to more competition and a larger number of equation overwrites per timestep, thus reducing overall diversity. At a density of 64%, mobility is extremely limited. Regardless, self-replication emerges consistently, with larger species conglomerations on average. Figure 3 shows several timesteps of a 64% density simulation run.

Figure 4 compares the original results from the 1% density runs with a new set of results from a similar 1% density simulation where the only difference was that the outputs of all molecubes were randomly generated values in the range $[0, 1]$. These values were regenerated at each timestep. These data show that self-replicating species can occasionally arise from inherent properties of the simulation itself. However, these species are on average the minimum possible size (two molecubes per conglomeration, the minimum number required to be defined as a conglomeration) and comprised of the minimum number of conglomerations (two conglomerations, the minimum number required to be defined as a species). Thus, while a minimal amount of self-replication can occur in the system by chance, having the genomes control the molecube outputs allows for a larger number of self-replicating species to emerge from the simulation. These genome controlled species are also on average more complex (i.e. more molecubes per conglomeration) and more reproductively viable (i.e. produce more copies of themselves) than their randomly arising counterparts.

Figures 5 and 6 show two examples of test simulations where replication was observed. In both scenarios, the test grid was 9x9x9 and all conglomerations were extracted from the same original 1% density simulation run. The conglom-

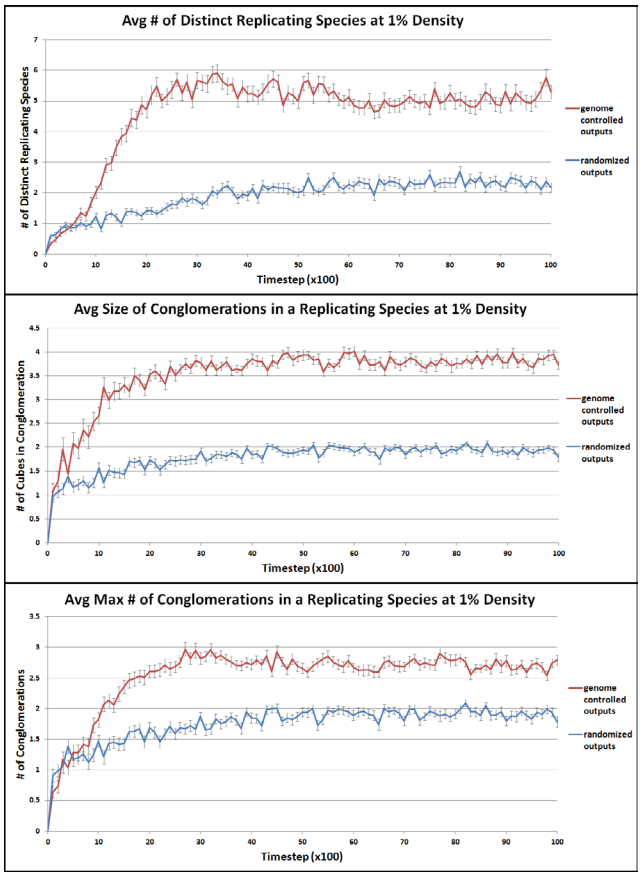


Figure 4: Experimental results for 1% density runs. "Genome controlled outputs" are the original results from the simulation as described. "Randomized outputs" are results from a simulation identical to the original, except that the outputs of each molecube were set to random values at each timestep. Each result is an average over 100 randomly initialized runs and error bars show standard error.

erations shown in Figure 5 were from timestep 7,900, while those in Figure 6 were taken from timestep 9,700. Figure 5 shows a large conglomeration dividing multiple times. It begins the test simulation composed of eight molecubes, which was its structure when it was extracted from the original simulation. It splits almost immediately into two groups, one of three cubes and one of four, leaving a single cube unused. The conglomeration of size four soon splits again into two groups of two cubes. One of these two groups attaches to a genetically distinct conglomeration of size two and after a few timesteps of back-and-forth stochastic genetic exchange, it is able to overwrite the foreign genomes with its own, thus becoming a genetically homogeneous conglomeration of size four. By the end of the test run, the original conglomeration of eight cubes has replicated multiple times, with the help of two cubes consumed from a foreign conglomeration.

In Figure 6, the blue conglomeration with four cubes consumes the two cubes in the green conglomeration. It then moves on to attach itself to the orange conglomeration. Despite being one cube smaller, the stochastic overwrites work in the blue conglomeration's favour, allowing it to rapidly overwrite the orange conglomeration. Finally, the single 13-cube conglomeration splits into two genetically identical conglomerations of size six, leaving a single cube unused. Thus in only 15 timesteps, the blue species was able to consume all of the molecubes in the test simulation and use this material to self-replicate.

It seems counter-intuitive that self-replication would arise in so few timesteps considering the large number of inputs and outputs for a molecube controller. In low-density situations, a self-replicating conglomeration must be able to move through the simulated 3D world and attach to new molecubes in ways that do not impede mobility. Moreover, at all densities, replicators must be able to detach at appropriate inter-cube connections and at appropriate times to produce viable copies. It turns out that a simple cube controller can produce these desired properties. For example, the controller in the blue cubes in Figure 6 is largely static, with the majority of its outputs set permanently to 0 or 1. This includes its turn output. Four of its six attach/detach outputs are static, with two set to 0 and two set to 1. The only fully dynamic outputs[1] are two of its attach/detach outputs, shown in simplified form in Equations 2 and 3.

$$dy_9/dt = dx_2/dt \tag{2}$$

$$dy_{11}/dt = \begin{cases} 0.074, & \text{if } x_7 = 0.0 \\ -0.77, & \text{if } x_7 = 0.5 \\ -0.54, & \text{if } x_7 = 1.0 \end{cases} \tag{3}$$

Thus, as in 2D cellular automata, a simple controller governing the interaction of multiple identical agents in a simulated 3D world can produce surprisingly complex behaviours. Note that the attach/detach output shown in Equation 2 depends on an incoming communication bit, demonstrating how communication bits can be used to decide when and where a cube conglomeration should split.

Conclusions and Future Work

As far as the authors know, the results presented in this paper are the first cases of the spontaneous emergence of self-replication in a simulated three-dimensional environment. Previous results (e.g. [Studer and Lipson (2005); Chou (1997)]) occurred in two-dimensional scenarios. Furthermore, by simulating molecubes that have been constructed in the real world, we are one step closer to evolved,

[1] "fully dynamic" outputs are ones that continue to change over time. This controller also had several "partially dynamic" equations that could change an output once before becoming static.

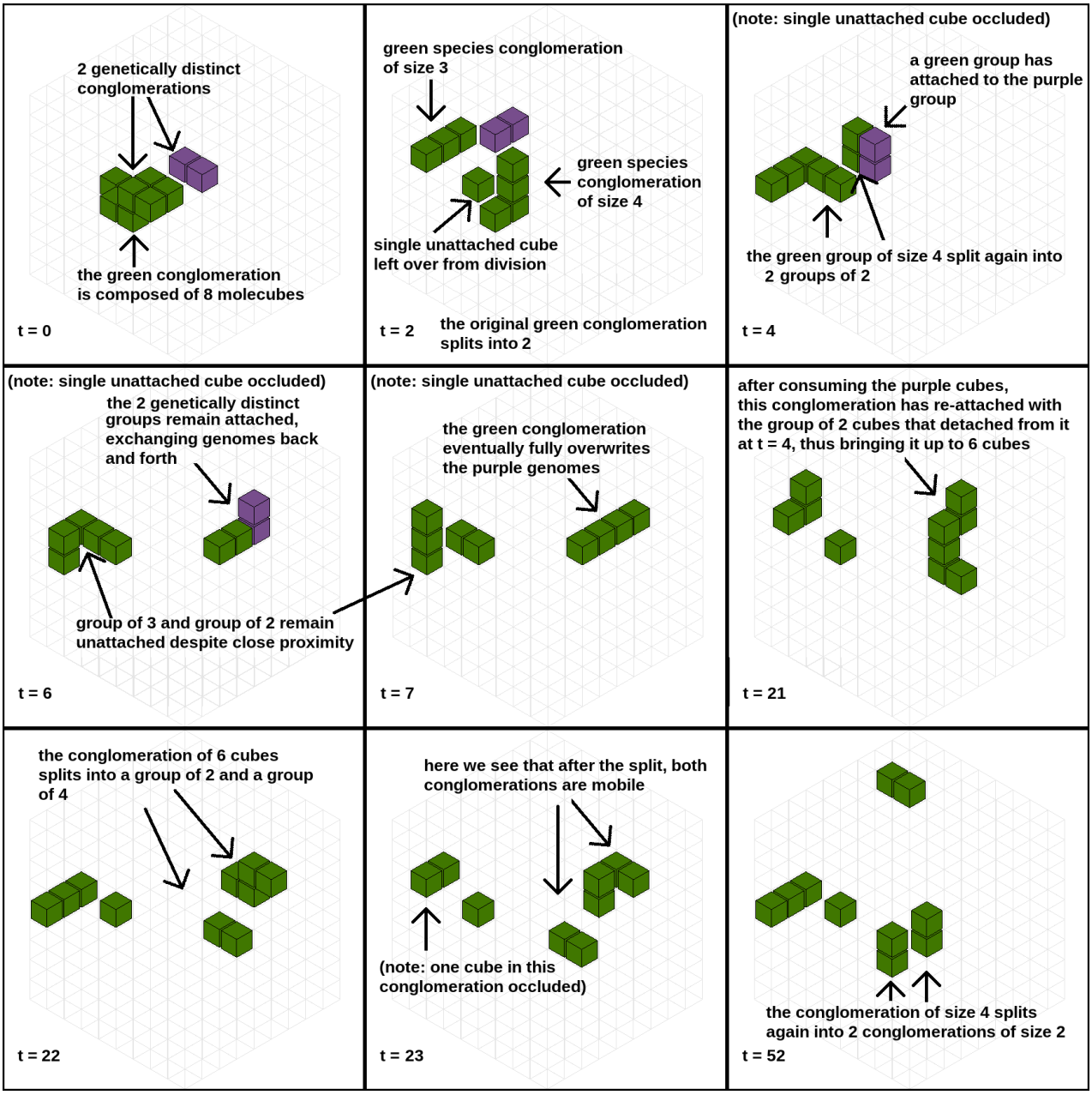


Figure 5: Test simulation using conglomerations from timestep 7,900. Colours represent relative genetic distance. A large conglomeration replicates multiple times. It also captures a small genetically distinct conglomeration and uses its cubes for self-replication.

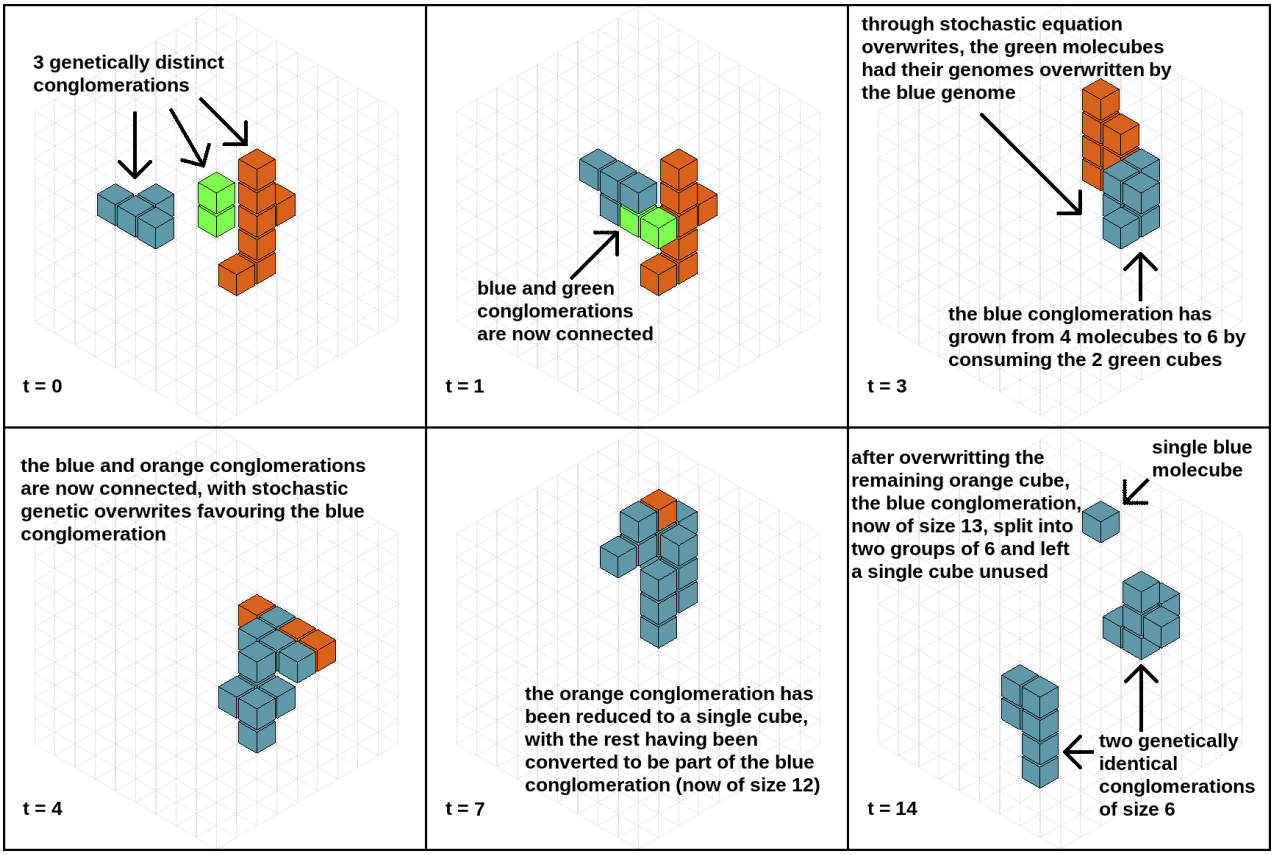


Figure 6: Test simulation using conglomerations from timestep 9,700. Colours represent relative genetic distance. The blue conglomeration consumes the other groups and uses their cubes to self-replicate.

physically realizable self-replicating machines. The next steps toward this goal would be to incorporate more physics into the simulation, including gravity, and to have the 3D simulated world be continuous instead of partitioned into a discrete grid.

In a 3D simulation, evolving controllers have a large number of inputs and outputs to contend with, and the number of potential situations in which a molecube conglomeration might find itself is very large. Future work should focus on further evolving these self-replicating species in an effort to produce species with more complex behaviours. Incorporating nature-inspired operations such as crossover and random death might help to increase the evolved capabilities of the controllers.

Despite the complexities associated with a three-dimensional world, a plethora of self-reproducing molecube conglomerations emerged in every run of our 3D simulation at densities of 1% and higher. Using simple, largely static controllers, these conglomerations were able to collect other molecubes and use them to produce new, genetically identical conglomerations. The simplicity of the controllers coupled with the frequency of the emergence of self-replication in scenarios requiring mobility as well as in scenarios that allowed for only limited mobility demonstrates that a diversity of surprisingly complex behaviours can emerge from the interactions of relatively simple agents in a simulated three-dimensional world.

Acknowledgements

This work was supported in part by the U.S. National Science Foundation's Office of Emerging Frontiers in Research and Innovation (grant number 0735953). Paul Grouchy would also like to thank the Natural Sciences and Engineering Research Council of Canada for its support through the Canada Graduate Scholarship and the Michael Smith Foreign Study Supplement.

References

Adami, C. and Brown, T. C. (1994). Evolutionary learning in the 2d artificial life system avida. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV*, pages 377–381, Cambridge, MA. MIT Press.

Bonner, J. T. (1998). The origins of multicellularity. *Integrative Biology: Issues, News, and Reviews*, 1(1):27–36.

Chou, H. (1997). Emergence of self-replicating structures in a cellular automata space. *Physica D: Nonlinear Phenomena*, 110(3-4):252–276.

Grouchy, P. and D'Eleuterio, G. M. T. (2010). Supplanting neural networks with ODEs in evolutionary robotics. In *Proceedings of the 8th international conference on Simulated evolution and learning*, SEAL'10, pages 299–308, Berlin, Heidelberg. Springer-Verlag.

Mytilinaios, E., Desnoyer, M., Marcus, D., and Lipson, H. (2004). Designed and evolved blueprints for physical self-replicating machines. In *in Proc. of the 9 th Int. Conf. on the Simulation and Synthesis of Living Systems (Artificial Life IX). MIT*, pages 15–20. MIT Press.

Neumann, J. V. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA.

Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd.

Ray, T. S. (1992). An approach to the synthesis of life. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, pages 371–408. Addison Wesley Publishing Company.

Studer, G. and Lipson, H. (2005). Spontaneous emergence of self-replicating, competing cube species in physical cube automata. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, late breaking paper, Washington D.C.*

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.

Zykov, V., Mytilinaios, E., Adams, B., and Lipson, H. (2005). Self-reproducing machines. *Nature*, 435(7039):163–164.