# Spontaneous emergence of self-replicating, competing cube species in physical cube automata

Greg Studer
Computer Science
Cornell University
Ithaca, NY 14850, USA
gms25@cornell.edu

Hod Lipson
Computational Synthesis Laboratory
Cornell University
Ithaca, NY 14850, USA
hod.lipson@cornell.edu

## ABSTRACT

We propose and implement a new variant of cellular automata incorporating physical rules, based largely on the nonuniform cellular automata type first investigated by Sipper [10]. The new automata rules are designed to be realizable in an actual electromechanical robotic module called a "molecube," which was built by us [6][14] and shown empirically to have self-replicating ability. The spontaneous and continuous emergence of simple self-replication is demonstrated using this automata platform. In addition, qualitative differences are observed between global state mutation and reproductive mutation.

## Keywords

Cellular Automata, Self-Replication

## 1. INTRODUCTION

Cellular automata have a long, storied history investigating basic notions of unpredictability, emergent behavior, and self-replication. Motivated in the 1940s by the question of how a machine could reproduce itself, von Neumann invented cellular automata and a complex cellular architecture capable of recreating any other automata structure, including its own [11], sparking early interest in computational reproductive systems [4]. This initial work has been refined by many other authors in the ensuing years: the complexity of the replicating structures was reduced dramatically [3], replicators were analyzed in terms of how information was used [5], and recently the spontaneous emergence of replicators in a cellular automata was demonstrated [2]. However, in one aspect the basic automata model used in all these researches remains unchanged; every cell in the automata operates according to exactly the same rules.

In a 1995 paper "Studying artificial life using a simple, general cellular model" [10], Sipper proposed a variation on the classic models he called nonuniform automata. These automata are populated by a grid of cellular "organisms," which are rule sets living in a cell that determine the next state of the cell and neighboring cells from the previous state of these cells. Organisms can move and replicate by copying their rules into neighboring cells. This small modification from the standard cellular automata creates remarkably complex behavior from simple, modular automata rules, including evolutionary effects similar to those observed in artificial life studies. By separating the organisms' local rules from the global rules which all organisms must obey, Sipper effectively created a virtual environment for virtual creatures in the well-understood automata paradigm. Nonuniform automata behavior in this way can be understood along the lines of program-based artificial life platforms such as Tierra [9] and Avida [13]. In Tierra and Avida, digital organisms consisting of instruction loops evolve without explicit fitness constraints over time. Placed in a virtual environment with memory and processing resources, the organisms are allowed to compete using their Turing-complete instruction set by copying themselves and increasing their execution time. Nonuniform automata organisms compete in an analogous auto-adaptive fashion because they are pressured to avoid overwrite by other organisms' rules. However, they are less complex. Instead of a Turing-complete basis, nonuniform organism rules are only finite automata, and thus can only perform less complicated operations individually. The entire automata space is Turing-complete, but the individual cells are not.

All of these platforms, Tierra, Avida, and nonuniform automata, produce intriguing results relating to self-replication and evolution, but they remain only an analogy to real-world systems. Many of the operations nonuniform automata and virtual organisms perform would be impossible in *any* system constrained by the laws of physics, such as replication into empty space, globally simultaneous operations, unconstrained structural changes, and monopolizing time from other organisms. More mechanical self-replication platforms have been described in literature by Penrose [8] and Chirikjian [1], but are rather inflexible as to their applications. The nonuniform cellular automata, however, is an incredibly general and simple abstraction, and at minimum provides the notion of interacting rules in *space* with *simple structures* of multicellular organisms. For this reason, it is closer to a physical simulation than many other artificial life simulations.

In this paper we propose and describe a type of nonuniform cellular automata that is physically realizable, based on an electromechanical cube which moves through magnetic interactions with other cubes and containing an actuator for motion. Real "molecubes" are described and built in our other research [6][14]. The original molecube paper [6] proposed a two-dimensional cellular automata simulation of these structures, and the work described in herein addresses that goal. The simulation remains discrete, but otherwise mechanical rules simulating conservation of mass through conservation of cubes and energy limitations create a more physical world to investigate emergent replication and the properties replicating structures may possess. While results are indeed preliminary, various small forms of spontaneous self-replication have already been observed, as well as surprising effects arising from how mutation is applied as a genetic operator. It is our hope that this system will allow us to investigate properties of motion relating to replication, as well as provide a basis for understanding the control and movement patterns necessary for replication of molecubes and arbitrary machines in the real world.

## 2. PHYSICAL CUBE AUTOMATA

The physical cube automata grid is a two-dimensional N × N grid with periodic boundaries. Every position in the automata may be occupied by exactly one cube, or is empty.

A cube in the automata is defined as a stateful, physical entity. This differs from much other work in cellular automata, but the cube automata was designed in this way so that the cubes could easily represent physical objects or organisms. Cubes in the automata cannot be created or destroyed, only moved and changed, thus providing a concept of conservation of mass.

The cube was designed for the automata as a two-dimensional, discrete simulation of an actual physical molecube [6][14], with electromagnets on four of its "side" faces (viewed from above). A cube's magnets can be turned on and off to allow a cube to attach to other cubes, and currently two "on" magnets are simulated to always attract. A diagonal cut is made from the top to the bottom face of the cube, separating the cube into two triangular prisms. This allows the cube to "swivel" its two halves with respect to one another in three dimensions (see Figure 2). In practice, this swiveling action is easily be implemented as an internal cube actuator. All cubes are

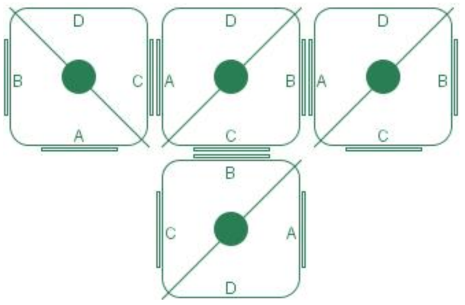

**Figure 1. Schematic top view of a sample cube structure where the letters on each side cube indicate the cube orientation. The swivel cut dividing the cube is shown by the diagonal line through each cube. Active magnets are shown by slim rectangles on the cube sides. Colored circles at the center of the cubes indicate a particular rule set.**

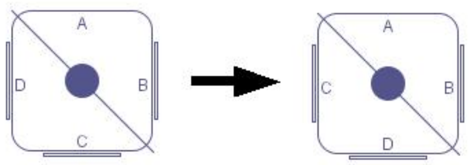

**Figure 2. A cube may swivel one half with respect to the other, based on the direction of the swivel cut.**

identical except for the direction of this swivel cut, which can either go from top-left to bottom-right or top-right to bottom-left (see Figure 1).

In addition, the cube state includes "software" controllers, which determine the next state of the cube magnets, which halves of a cube should swivel, and whether the cube should overwrite the controllers of its neighbors. Every controller also has an associated color, which is chosen arbitrarily and used when graphically representing the cube automata. Controllers directly correspond to the organism rules in the standard nonuniform cellular automata, but are separated out into their logical roles for clarity. While molecubes could in theory have Turing-complete control systems, the emphasis is initially placed on simpler control schemes.

As input, each of the controllers receive four binary bits indicating which of the four neighbor cells are filled by a cube (von Neumann neighborhood). This type of input was chosen because molecubes are simpler to build with inputs on the cube faces. The magnet controller outputs four bits indicating the new state of each magnet, while the swivel controller outputs four bits indicating whether each of the cube halves should swivel. Because there are two possible swivel cuts through the cube, only two of these bits are used in a particular cube. The other two bits are used when the controller is copied to a cube with a different direction swivel cut, allowing the rules to specify independent behavior for the two cube types. Finally, the overwrite controller outputs four bits indicating which neighbors A,B,C, or D should be overwritten. A cube will overwrite its neighbors' controllers, replacing all of them with its own versions, if the corresponding neighbor output bit is 1. Overwrite contention between two cubes to overwrite the same cube or each other is resolved by whichever cube executes first, as explained below. This is inherently a random choice, so this automata is somewhat nondeterministic. This may allow us to relate our results to other work with stochastic molecube interactions on a 2-D substrate [12]. Stochastic interactions are applicable especially to smaller-scale molecubes which may not be independently powered.

The function of the software controllers is essentially to map binary strings to other binary strings, so many different implementations were available to us. Sipper's nonuniform automata implementation used finite automata, but at this time, we are using a binary decision tree variant, which is computationally equivalent. A controller decision tree consists of a tree of nodes containing values indicating which input bit to use when deciding the next branch. Between nodes, <output value, output bit position> pairs may be emitted, and when a leaf node is reached all output values must have been specified. This representation is useful because it is able to represent any symbolic input ?b output mapping, is easy to use when generating randomized controllers, and lends itself well to testing because it can be readily understood as a series of if-statements. Also, in the future, controllers could easily be combined with one another by merging trees together at random nodes.

Cube magnets are able to connect to one another if they both have adjacent magnets in an "on" state. We refer to these connected cubes as cube structures. Two cube structures are

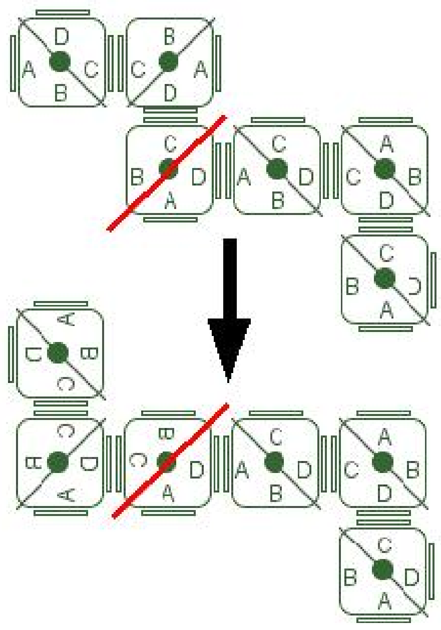

**Figure 3. Sample cube swivel operation, around the cut indicated in red. The B/C side of the cube is swiveled, resulting in the attached cubes changing position and orientation.**

considered equivalent if the magnet connections create the structures determine the same graph, i.e. if the magnets connect the cube structures the same way, taking magnet letters into account. In this way, the structure is independent of the orientation of the cubes or swiveling, because these factors do not affect magnet connections.

When a cube in a structure swivels a connected magnet, the adjacent cube connected to that magnet stays attached. All cubes attached to that adjacent cube also stay attached, and so on. The intuition is that all attached cubes to a particular magnet create a type of cube "arm," which swivels around in the third dimension and assumes new positions in the automata afterward, in a flipped orientation. Swiveling is only allowed, however, if the swiveled cube arm would not collide with other cubes in the swiveled positions and if the arm not also attached to the other swivel side of the cube. As an example, in Figure 3 the swivel would not be allowed to occur if the cube arm attached to magnet C was also attached to magnet A, because then the swivel would need to break a magnet connection. Through this swiveling action, cube structures are able to move themselves around the automata. It should be noted, as seen again in Figure 3, that the cube cut direction remains unchanged by swiveling, even after flipping the arm orientation.

Initially, the cells of the cube automata are populated with a probability that the cell will contain a cube. In this way, cube probability corresponds roughly to cube density. In our experiments, cube probability is kept at 10%, mostly because it yields interesting results. Higher density makes it difficult for the structures to move, while lower density makes it less likely that structures would interact. Initial cubes are generated with

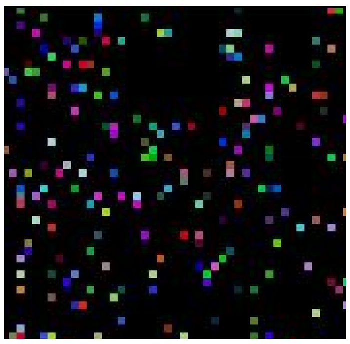

**Figure 4. Example portion of an initial automata state. Empty cells are represented by black regions, while cells containing cubes are represented by the average color of the controllers of that cube.**

random cut direction and randomly generated controllers. The initial cubes are also assigned a randomly generated cube ordering.

We execute the cube automata by continually looping through the cube ordering to find the next cube, gathering the neighbor cube input, and then executing the output of that cube's controllers in response to that neighbor input. A full iteration or "time step" has passed when all the cubes have executed once. This differs from the simultaneous execution of the standard nonuniform cellular automata, but is a necessity in the cube automata because of the possibility of collisions occurring between swiveled cube arms. These collisions would be difficult, if not impossible to resolve if they all occurred simultaneously.

## 3. PRELIMINARY EXPERIMENTS

The behavior of the cube automata is determined by the cube controllers, and the number of unique controllers for a binary four input to four-output mapping is 16^16 ˜å 1.84 × 10^19, yielding a large possible range of controller behavior. In these initial experiments with the cube automata, the goal was to discover controllers which would create self-replicating cube structures, which has been demonstrated as possible in [10].

Because a given initialized system will not necessarily exhibit replicating behavior, random mutations in the automata continually search the space of possible controllers. It may appear at first that these random mutations are searching the large space of controllers in a totally unsystematic manner, but surprisingly replicating structures were found a few hundred iterations from the initial state. These replicating controllers emerge because controllers which are effective in forming structures to move and overwrite other cubes are less affected by mutations killing off members of their "species" over time. (For brevity, cubes in the automata which contain the same controller functions will be referred to as being of the same species). In effect, mutations continually cull the less active species from the automata, resulting in a natural selection pressure.

The way mutations were applied in our experiments took two forms, global mutations over the entire cube state, and mutations linked to cube overwrites. Initially, it was assumed that the two types of mutations would produce similar behavior, as was observed by Pargellis in his study of self-replicating digital organisms [7]. However, in our automata, the two different mutation logics produced significantly different results, and are presented separately below.

## 3.1 Global Mutations

In the first set of experiments, mutations were applied globally to the automata state. Every iteration, each cube had a 0.5% chance of being mutated and having its controller set replaced by new random rules. For example, in a 100 × 100 grid with 10% cube probability when initializing the automata, five cubes on average are mutated each iteration.

At first, only a single type of stable structure arose immediately, a growing conglomerate of connected chunks of cube species. The connected chunks could repair themselves against mutation by continually overwriting their neighbors, and so tended to survive longer than other species which did not form large structures. This was observed as "survival of the flattest" in the Avida paper [13]. Eventually, all the conglomerates grouped together into one large semi-static final structure.

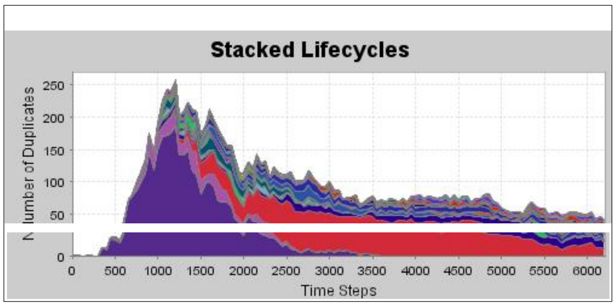

**Figure 5. Species replication emerging from a sample execution of the globally mutated cube automata for 6000 iterations. Automata size was 200 × 200 cells, and the automata was sampled every 50 iterations. The different colors correspond to the different cube species, with newer species emerging at the top of the stack. The number of duplicates was calculated by determining the structures consisting only of a particular species, and then counting all the structures which had a copy somewhere else in the automata.**

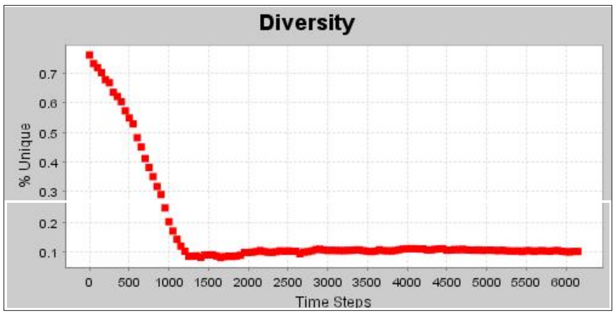

**Figure 6. Cube diversity during sample automata execution with global uniqueness expressed by (number of species) / (number of cubes).**

Because we were interested in self-replicating behavior, this stable structure capturing all the cubes had to be discouraged. For this purpose, a new "swivel arm limit" rule was created that does not allow a cube to swivel if more than 10 cubes were attached to the swiveling portion. By "freezing" large structures in place so that their component species are less effective in overwriting large numbers of cubes, this rule was quite effective in limiting the growth of huge structures. Because the same problem occurred with overwrite-linked mutations, described further down in the paper, the rule was kept in effect for those experiments as well.

With global mutations and this structure size limitation, two types of semi-stable structures emerge. As can be seen in Figure 5, the first purple species creates large numbers of duplicated structures which emerge to quickly dominate the early automata. These structures are very mobile, and so are able to overwrite large portions of the board in a short amount of time. After a few thousand iterations, however, fast duplicates are overtaken by larger static structures (see Figure 9) which slowly overwrite cubes from the moving species. This brings much of the movement to a halt after many iterations and the automata into a semi-steady state. Replicated structures still remain after long periods of time, but these structures are not very active and therefore unstable under global mutation. As enforced by the swivel arm rule, the larger static structures are unable to move in their entirety, but they are able to move small outer arms of

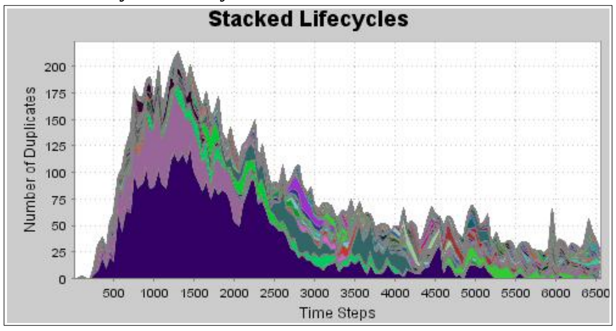

**Figure 7. Species replication emerging from a sample execution of the overwrite-mutated cube automata for 6000 iterations. Automata size was 200 × 200 cells, and the automata was sampled every 50 iterations.**

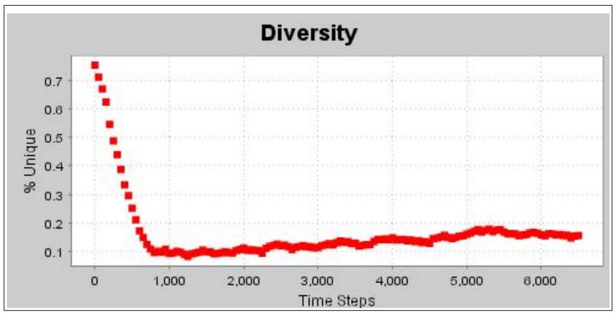

**Figure 8. Cube diversity during sample automata execution with overwrite mutations.**

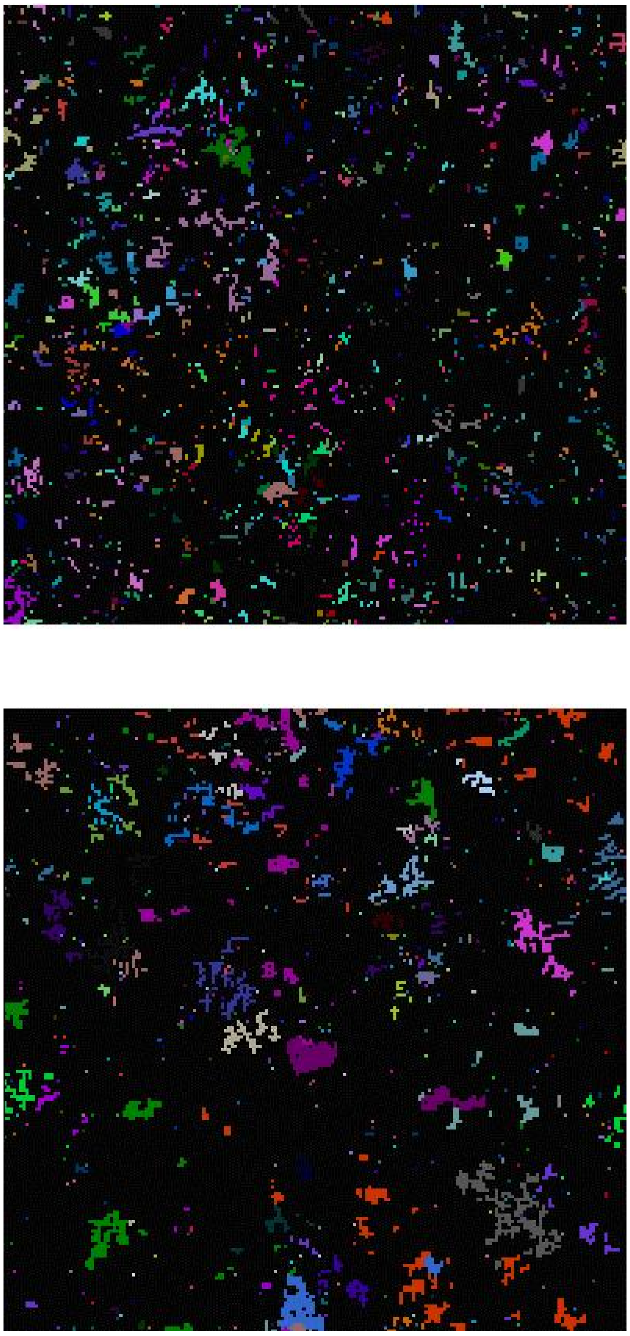

**Figure 9. Sample automata state using global (bottom) and overwrite (top) mutation after 6000 iterations. Note the large static structures using most of the cubes in the globally mutated automata, while the overwrite-mutated automata remains large structure-free.**

attached cubes to overwrite other cubes as they come nearby, somewhat like sea anemones capturing prey.

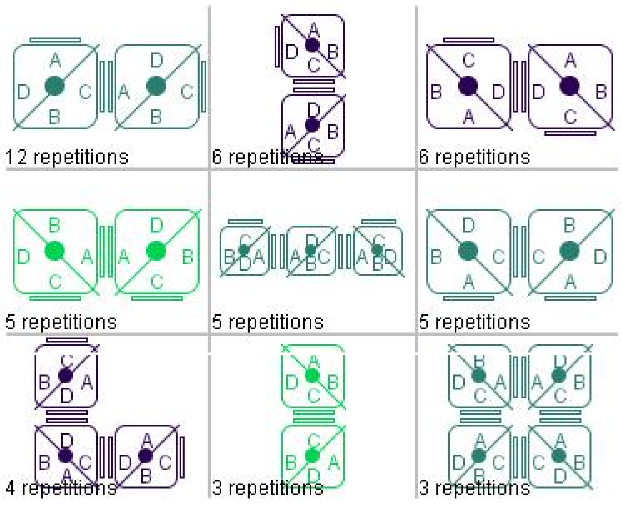

**Figure 10. The nine most common duplicated structures in the sample overwrite mutation automata at approximately iteration 3700. Note that the larger structures are often composed of connections similar to those in the smaller duplicated structures of their species. These species correspond to some of those displayed at iteration 3700 of Figure 7.**

## 3.1 Overwrite-linked Mutations

Inspired by natural mutations and the Pargellis paper [7], another mutation scheme to search the controller space was investigated: overwrite-linked mutation. In the natural world, mutations to species occur during replication, not randomly during the lifetime of an individual. In these experiments, global mutation did not occur, but instead overwrites had a 0.5% chance of failure, creating a neighbor cube with randomly generated controllers instead.

As was mentioned above, the initial globally mutated automata is very dynamic, but static clumping invariably occurs at later iterations. This clumping is avoided entirely with overwrite-linked mutation, because self-overwrites to “heal” mutated structures actually encourage more mutation to occur. What results is a continually dynamic automata of many new duplicated species structures. Even over large numbers of iterations, the automata space has not been observed to slow production of new duplicated species. This seems a paradoxical observation, because performing mutations on overwrites actually allows automata states which are entirely stable. If no overwrites can occur in a particular sequence of automata states, then that sequence will never change. This has never been observed in the limited experiments we have performed, but it may simply be an unlikely situation that rarely occurs.

## 4. DISCUSSION

There are two possible explanations for the duplicated cube structures in the automata: a structure factory or self-replicating cube structures. By structure factory, we mean a set of cube structures that, while not producing copies of themselves, are

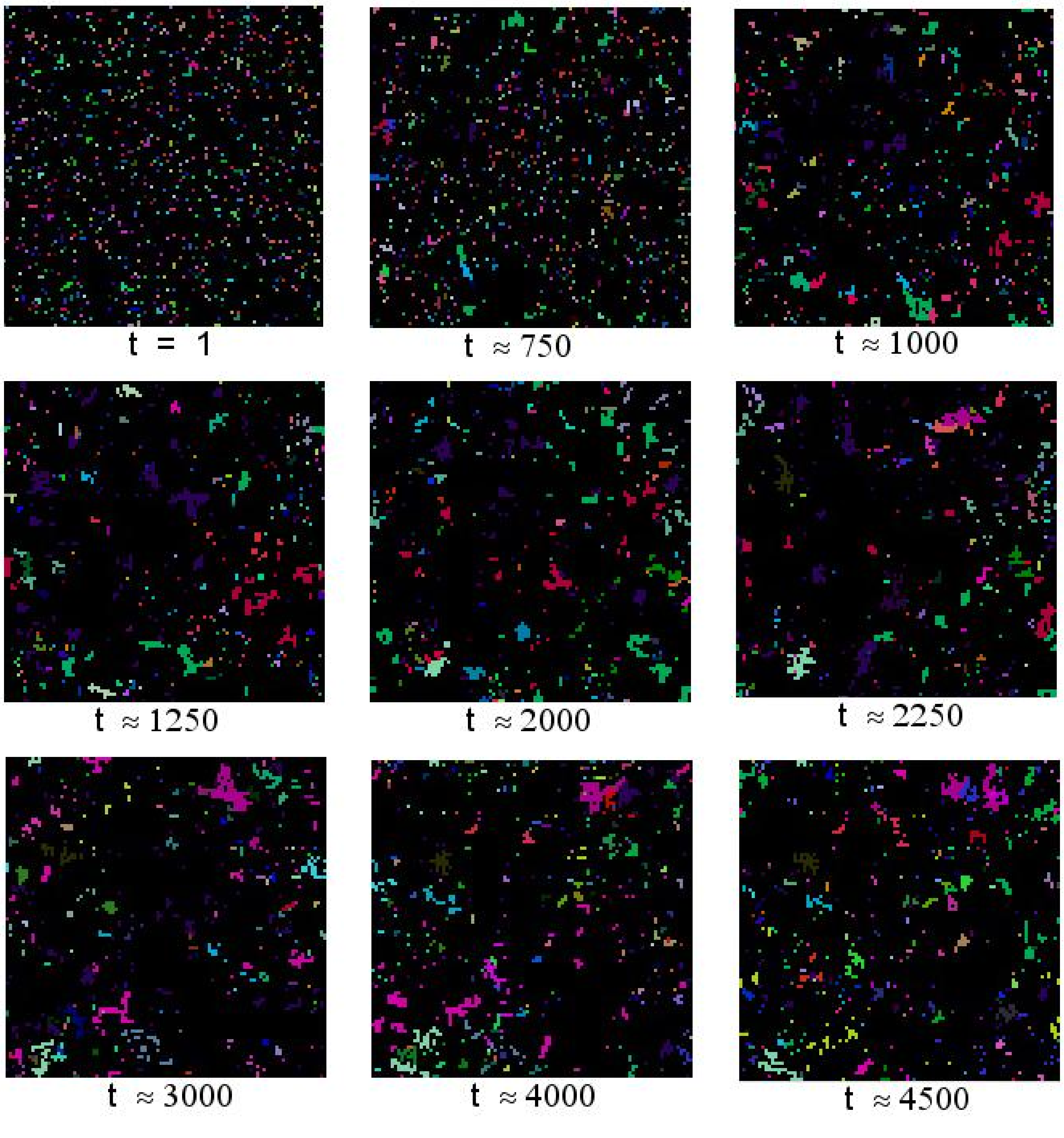

**Figure 11. Another sample execution of the cube automata using overwrite mutation. Note the emergence of a self-replicating purple structure at t ≈ 750, which expands to overwrite much of the board at t ≈ 1250. At t ≈ 2000, new green and red structures emerge which compete for cubes, and at t ≈ 2250, a pink structure emerges and eventually replaces the purple replicators at t ≈ 4000. At t ≈ 4500, the purple replicators have new competition from yellow, green, and light blue structures.**

able to produce copies of a different type of cube structure. These are the only two ways in which duplicated structures can arise in the automata, essentially by definition. All mutations create a single unique cube species as the origin of all later cubes of that species, thus a single cube creates these factories or replicators through overwrites, and so demonstrates self-assembly in some limited form (though this could be, and probably is in almost all cases, highly dependent on the cube immediate surroundings).

Such cube factories are visible especially in the global mutation experiments, because large self-reinforcing structures are stable using these rules. This can be seen in Figure 5 as the pinkish-orange duplicated species that slowly dies over time. This pink duplicated structure was not particularly mobile, but it was duplicated anyway because there were larger pink structures (the remains of which are still visible in Figure 9) supporting the duplication. To create new structures, the larger, immobile pink factories required moving structures to come close enough to overwrite them, and as iterations went on more and more automata structures were of the large, immobile type.

The purple duplicated species in Illustration 5 underwent explosive growth of over 100 duplicates in the 600 to 800 iteration period, when on average a duplicate was created every two iterations. No large purple factories existed at this early time step. It is improbable that a mobile single factory created this many duplicates, or that multiple unique factories were created without using cubes from the duplicated structures (if cubes from the duplicated structures were used to create more factories, then the structures are performing self-replication). It is probable, then, that between 600 to 800 iterations the purple structures were exhibiting some form of self-replication. This is supported by observation of the automata in progress, as illustrated by Figure 11. Small, two-and-three cube replicated structures travel outward from a central location, creating geometrically more copies of themselves on the growing border between the area they have consumed by overwrites and the rest of the automata.

With overwrite-linked mutation, factory structures are necessarily unstable, because all overwrites run the risk of pathological mutation. Any particular structure or set of structures that continually performs overwrites (such as a structure factory) must eventually succumb to these mutations, and a structure cannot heal itself with self-overwrites without being even more susceptible to harm. This makes the emergence of different types of self-replicating species more common, as can be seen from the overwrite diversity graphs (Figures 6 and 8). Larger numbers of replicating species also mean that these species must compete for cubes, and this can be seen in the smaller number of duplicates which coexist simultaneously using overwrite mutation rules in Figure 7. The geometric growth of duplicated structures indicating self-replication can still be observed, however. For example, the blue structures between 600 and 650 and iterations create over 40 duplicates (Figure 7), an especially large number considering how unstable duplicating structures are.

## 5. CONCLUSION

The spontaneous and continuous emergence of small self-replicating cube structures has been observed in the cube automata. Our automata platform could be realized using physical molecubes, so the replication observed would actually occur in a real test using 2-D molecubes if constrained to a discrete grid. Obviously this work is very preliminary, and in the future we hope to address more precisely how the motion of structures directed by controller rules allow structures to self-replicate. Most of the duplication that occurs currently is of the smallest possible two-cube structures, though larger duplicated cube combinations have occurred as in Figure 10. This may be because the search for replicating controller species is directed, but still based only on random mutation, so it would be interesting to see whether larger replicating species could be discovered through recombining or incrementally mutating controllers as was done in [10].

The controllers as currently implemented are also limited in their behavior by the input that they receive, which is only four bits indicating the presence of surrounding cubes. More interesting controllers may be evolved if the input is richer. Inter-cube communication using various numbers of communication bits might yield more coordinated cube motions, and adding internal cube state would allow cubes to “remember” portions of their surroundings. These extra sources of information make the controller space much larger, and it may take longer to evolve very successful replicators, but these are traits real replicating objects often use to their advantage.

## 6. REFERENCES

[1] Chirikjian, G. S. Zhou, Y., and Suthakorn, J. Self-replicating robots for lunar development. *IEEE/ASME Transactions on Mechatronics*, 7, 4, pp. 462-472, 2002.

[2] Chou, H. and Reggia, J. A. Emergence of self-replicating structures in a cellular automata space. ***Physica** D*, 110, No. 3-4, pp. 252-276, 1997.

[3] Codd, E. F. *Cellular Automata*. Academic Press: New York, 1968.

[4] Freitas, R. A., Merkle, R. C. *Kinematic Self-Replicating Machines*. Landes Bioscience.

[5] Langdon, C. G. Self-reproduction in cellular automata. *Physica D*, 10, pp. 135-144, 1984.

[6] Mytilinaios, E., Desnoyer, M., Marcus, D., and Lipson, H. Designed and evolved blueprints for physical self-replicating machines.

[7] Pargellis, A. N. The spontaneous generation of digital life. *Physica D*, 91, pp. 86–96, 1996.

[8] Penrose, L. S. Self-reproducing machines. *Scientific American*. 206, 6, pp. 105-114, 1959.

[9] Ray, T. S. An approach to the synthesis of life. *Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity*, 11, pp. 371–408.

[10] Sipper, M. Studying artificial life using a simple, general cellular model. *Artificial Life Journal*, 2, No. 1, pp. 1-35, 1995.

[11] von Neumann, J. *Theory of Self-Replicating Automata*. University of Illinois Press: Urbana, 1966.

[12] White, P., Zykov, V., Bongard J., and Lipson H. Three dimensional stochastic reconfigurations of modular robotics. Proceedings of Robotics Science and Systems, MIT, Cambridge, MA, June 8-10, 2005. [preprint draft

[13] Wilke, C. O., Wang, J. L., Ofria, C., Lenski, R.E., and Adami, C. Evolution of digital organisms at high mutation rates leads to survival of the flattest. ***Nature***, 412, pp. 331-333, July 19, 2001.

[14] Zykov, V., Mytilinaios, E., Adams, B., Lipson, H. Self-reproducing machines. Nature, 435, No. 7038, pp. 163-164, 2005.