

# Spontaneous Emergence of Self-Replicating Structures in Molecube Automata

Gregory Studer and Hod Lipson

Computational Synthesis Lab, School of Computer Science,  
Cornell University, Ithaca, NY 14853, USA  
gms25@cornell.edu

## Abstract

We propose and implement a discrete, two-dimensional evolutionary simulation of large numbers of interacting robotic modules called “molecubes,” which were shown empirically to have self-replicating ability (Zykov et al. 2005). In this simulation, the spontaneous and continuous emergence of large numbers of simple self-replicating molecube structures is observed without any explicit selection for this behavior. A quantitative comparison of structures’ self-replicability is investigated using a universal self-replication metric (Bryant and Lipson 2003), as well as the existence of interacting structure groups. Because of the discrete nature of the simulation and simple interactions, it bears strong resemblance to nonuniform cellular automata (Sipper 1995).

## Introduction

Modular robots have become a popular area of research in the development of adaptable robotic systems, especially when the robots are being designed for self-replication. Because structures built from multiple robotic modules are often able to move the individual modules they are made from, in a supportive environment these robotic structures can create copies of themselves. Current replicating designs are not very practical, but the promise of a more flexible replicating system remains enormous. In theory, self-replicating robots could colonize the moon and solar system (Frietas 1981), as well as revolutionize manufacturing here on Earth.

While mechanical self-replication was originally studied using tumbling blocks (Penrose 1959), current robotic replicating systems are more deterministic and complex. The recent first example of robotic self-replication, demonstrated in a semi-autonomous modular robot, involved a multi-step linking and assembly process (Chirikjian et al. 2002). More recently, a robotic structure composed of identical modules also performed replication (Zykov et al. 2005), adding the ability to form arbitrary 3D structures. In the future, it seems likely that mechanical self-replication technology will continue to improve, not only in large-scale robotic technology but also in micro-and nano-scale devices.

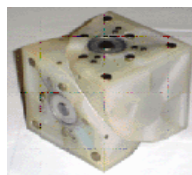
To design and build these future mechanical replicators, we need to understand how self-replication can arise in a mechanical system. Since the invention of

cellular automata in the 1940s (von Neumann 1966), researchers have been investigating the computational nature of replication (Langdon 1984). Originally this work was intended to study abstract systems, but computers have allowed more detailed simulations of replicating virtual organisms (Ray 1991, Wilke et al. 2001, Pargellis 1996), virtual chemicals (Hutton 2002), and virtual nucleotides (Smith 2002). Spontaneously emerging, self-replicating structures appear in many of these systems, but are robotic systems limited by macroscopic mechanical rules capable of supporting similar behavior?

To better understand the limits and potential of replicating robots, we propose and implement a simulation of modular robotic “molecubes” in a nonuniform cellular automata variant (Sipper 1995). Inspired by other artificial life and cellular automata simulations which observe spontaneous self-replication (Chou and Reggia 1997, Pargellis 1996, Hutton 2002, Smith 2002), we created an undirected evolutionary environment in which mechanical self-replication could evolve naturally over time. The rules of the simulation are based off the discrete mechanics and capabilities of real molecubes, which have been shown capable of preprogrammed replication in other research (Mytilinaios et al. 2004). We show using this simulation that multi-modular robotic replicators emerge spontaneously and continuously in simulations of large numbers of molecubes, despite the requirements of constant mass and mechanical motion.

## Molecubes

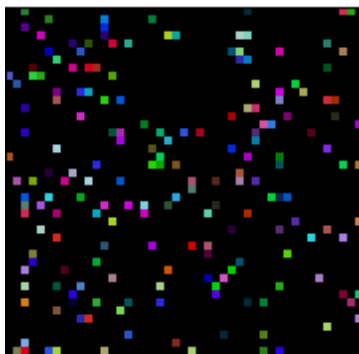
Designed in previous work at the Computational Synthesis Lab, molecubes are electromechanical cubes which are able to attach and detach from one another using electromagnets on the cube faces (Figure 1). Each molecube contains an actuator for motion, allowing one half of the cube to swivel with respect to the other half. Molecube structures of multiple cubes are able to position themselves in arbitrary, three dimensional ways using this swiveling action.



**Figure 1. Image of a real molecube. Notice the cut through the diagonal of the cube, allowing the two halves to swivel. (Zykov et al. 2005).**

## The ATOMM Simulation

The ATOMM (AuTomatic Organization in Mechanical Molecubes) simulation takes place on a two-dimensional  $N \times N$  cellular automata grid with periodic boundaries. Every position in the automata may be occupied by exactly one simulated molecube, or is empty.



**Figure 2.** Example portion of an initial automata state. Empty cells are represented by black regions, while cells containing cubes are represented by the color of the controller of that cube.

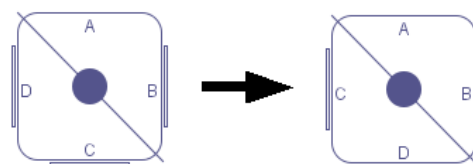
### Simulated Molecubes

A molecube (or ‘cube’) in the automata is defined as a permanent, individual entity. This differs somewhat from traditional work in cellular automata which does not explicitly identify elements other than automata cells. The cube automata has this additional requirement so that simulated molecubes easily represent actual robotic molecubes. As in reality, simulated molecubes are not created or destroyed in the automata, only moved and reprogrammed, leading to an overall conservation of cube mass.

The simulated behaviors of a cube are a two-dimensional, discrete simulation of real molecube behavior. Each cube has electromagnets on its four “side” faces (viewed from above), and these magnets can be turned on and off to allow a cube to attach to other cubes. Two “on” magnets are simulated to always attract, there is no simulated polarity. 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 3). In real molecubes, this swiveling action is implemented using an internal cube actuator. All cubes in the simulation are 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.

### Molecube Controllers

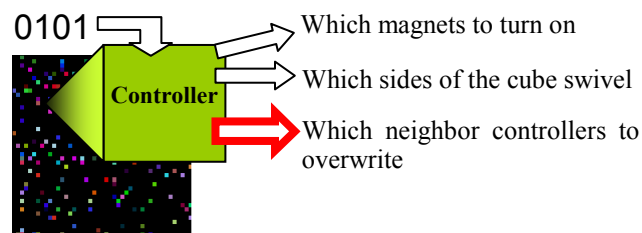
Every cube in the automata also has an associated “software” controller, which determines the next state of the cube magnets, which halves of a cube should swivel,



**Figure 3.** A cube may swivel one half with respect to the other, based on the direction of the swivel cut.

and whether the cube should overwrite the controllers of its neighbors. Each controller has an associated color, which is chosen arbitrarily and used to graphically represent the cube automata, as in Figure 2. This controller directly corresponds to the organism rules in the standard nonuniform cellular automata. Though our emphasis is initially placed on simple control schemes, real molecubes could be built with a Turing-complete control system, i.e. a microchip with memory.

Every iteration, a molecube controller receives four binary bits of input, indicating which of the four neighboring cells are filled by a cube (von Neumann neighborhood), and produces binary output to control the cube (Figure 4). The von Neumann neighborhood was chosen because real molecubes are much simpler to build with inputs on cube faces. For clarity, the cube controller is broken up into three logical sections: magnet controller, swivel controller, and overwrite controller. The magnet controller outputs four bits indicating the new on/off state of each magnet A, B, C, and D. A cube can be flipped and moved in the automata, so the magnets are labeled with letters to emphasize that a particular magnet is not always pointing in a particular direction. 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 to overwrite. A cube will overwrite a neighbor cube’s controller, replacing the entire neighbor controller with its own controller, if the overwrite controller output bit for the magnet the neighbor is adjacent to is 1. For example, if the A output bit is 1 and there is a neighbor



**Figure 4.** Schematic diagram of the role of a cube controller on a cube in the automata.

cube adjacent to magnet A, that neighbor's controller will be overwritten. There is never any overwrite contention between two cubes because the cubes execute one-by-one in series, as explained further below.

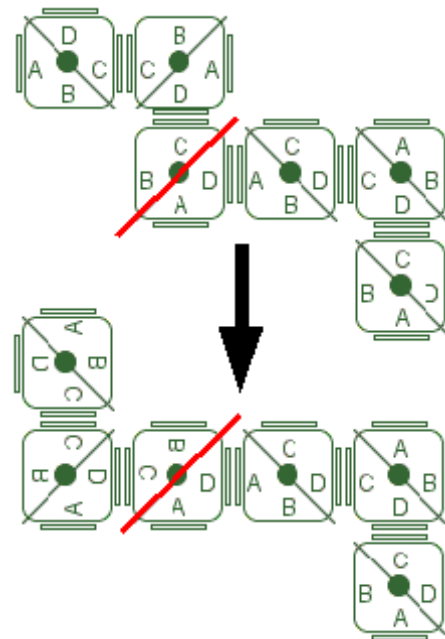
The function of the cube controller is essentially to map binary strings to other binary strings. We could have used many different controller implementations to accomplish this. In Sipper's nonuniform automata, finite automata were used as the controller. For simplicity we use a binary tree variant instead that can emit symbols on branches. 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 particular implementation is useful because it is able to represent any symbolic input  $\rightarrow$  output mapping, is easy to use when generating randomized controllers, and lends itself well to testing because it can be readily understood as a nested series of if-statements. Also, in the future, controllers could easily be combined with one another by merging trees together at random nodes.

### Cube Structures

Neighboring cubes are able to connect to one another if on or the other has an adjacent magnets in an "on" state. We refer to these connected cubes as cube structures. Two cube structures are considered equivalent if the magnet connections which 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 and 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 cubes attached to a particular magnet create a type of cube "arm," which swivels around in the third dimension and assumes a new position in the automata afterward, with flipped orientation (see Figure 5). Swiveling is only allowed, however, if the swiveled cube arm would not collide with other cubes in the new position and if the arm is not also attached to the other swivel side of the cube. As an example, in Figure 5 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 multiple swivels, cube structures are able to move themselves around the automata. It should be noted, as seen again in Figure 5, that the cube cut direction remains unchanged by swiveling, even after flipping the arm orientation.

The simulation also enforces that cube swivels move only a limited number of attached cubes. In our experiments, the limit was set to 10. If more than 10 cubes would be moved by a swivel, the swivel does not occur. Real molecubes have only finite power and are not able to



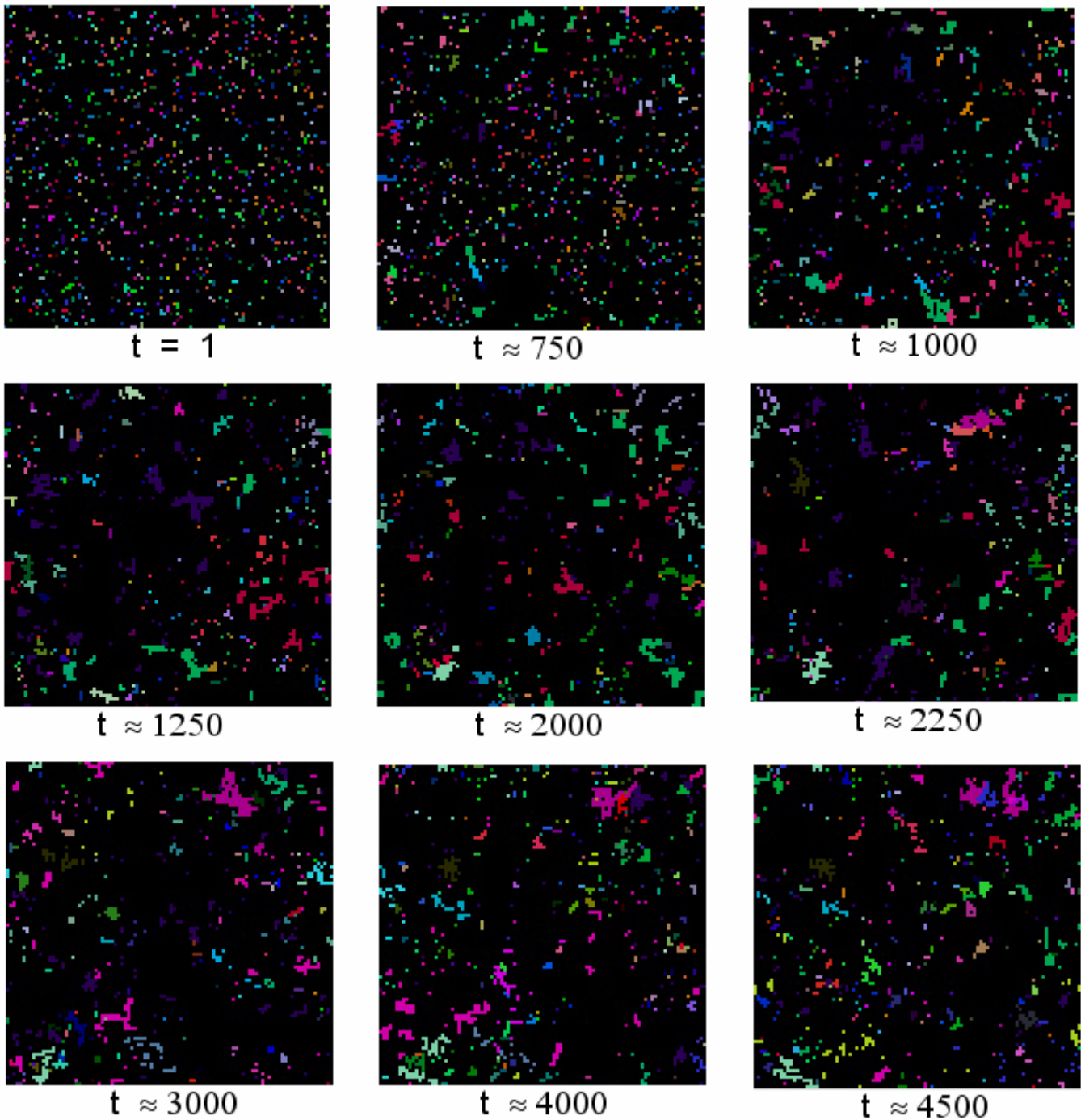
**Figure 5. Sample cube swivel operation, around the cut indicated in red. The B/C side of the cube is swiveled, resulting in the attached cubes changing position and orientation.**

move an arbitrary number of attached neighbors, so simulation reflects this fact. In a deeper sense, however, it was found that this rule is necessary to allow self-replication to emerge. With unlimited cube power, replicated structures do not emerge over time, as was observed in early experiments (Studer and Lipson 2005).

### Simulation Execution

Initially, the cells of the cube automata are populated with a probability that a cell will contain a cube (Figure 2). This 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 will interact. Initial cubes are generated with random cut direction and randomly generated controllers. The initial cubes are also assigned a randomly generated cube ordering.

The ATOMM simulation is executed 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 cellular automata, but is necessary in the cube automata because of the possibility of collisions can occur between swiveled cube arms. These collisions would be difficult to resolve in a realistic way if they all occurred simultaneously.



**Figure 6.** A sample execution of the cube automata for 4500 iterations. Note the emergence of a self-replicating purple structure at  $t \approx 750$ , which expands to overwrite much of the board at  $t \approx 1250$ . At  $t \approx 2000$ , new green and red structures emerge which compete for cubes, and at  $t \approx 2250$ , a pink structure emerges and eventually replaces the purple replicators at  $t \approx 4000$ . At  $t \approx 4500$ , the purple replicators have new competition from yellow, green, and light blue structures.

To simulate an open-ended evolutionary process in the automata, random mutations occur and continually search the space of possible controllers. These mutations occur on overwrites. With a small probability, set to 0.5% in our experiments, an overwrite will fail and the neighbor

controller will be replaced by a new, randomly generated controller.

The behavior of the cube automata is determined by the cube controllers, and there are many controllers possible. The number of unique controllers for a binary four input to

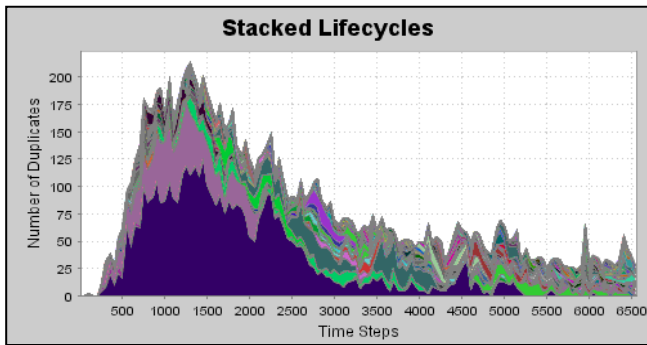


Figure 7. Species structure duplication emerging from a different sample execution of the cube automata for 6000 iterations. The automata size was larger,  $200 \times 200$  cells, and the automata was sampled every 50 iterations. The different colors correspond to the different cube species, with newer duplicated species emerging at the top of the stack. The number of duplicates was calculated by determining the homogenous structures consisting only of one particular species, and then counting all the structures which had a copy somewhere else in the automata.

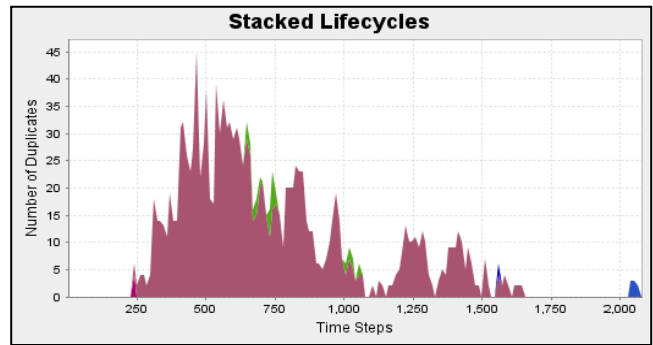


Figure 8: Species replication emerging from a sample execution of the automata displayed in Figure 9. Automata size was  $100 \times 100$  cells, and the automata was sampled every 12 iterations. There are much fewer types of duplicates because the automata size is smaller, and because this execution was chosen to analyze a single type of structures' replication.

four-output mapping is  $16^{16} \approx 1.84 \times 10^{19}$ , and each simulated controllers have three of these sub-controllers. In these experiments with the cube automata, the goal was for evolution to discover controllers in this large search space that would support self-replicating cube structures. It was initially not clear that such controllers existed. While self-replication was shown to be possible, the motions require coordinated controllers with richer input.

## Results

It may seem at first that random mutations would search the large space of controllers in a totally unsystematic manner, but duplicated structures are often observed a few hundred

iterations from the initial state (see Figure 6 and Figure 7). For simplicity, only homogenous structures containing a single controller type were analyzed for duplication. It is possible that non-homogenous duplicated structures are created, but none were easily observed. In general, homogenous duplicates appeared in large numbers, and the goal of the work was to search for the existence, not the extent, of self-replicating structures.

These duplicated structures 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).

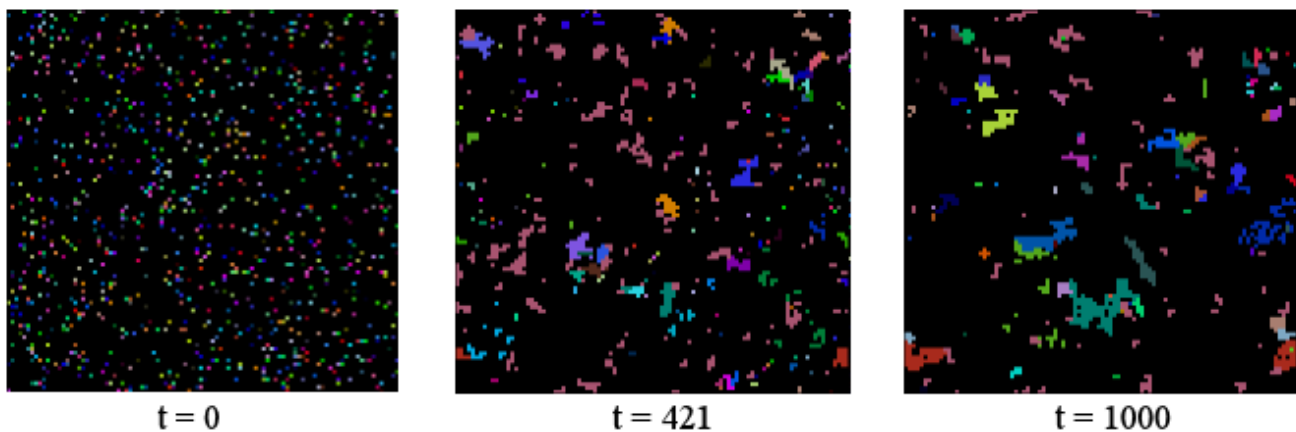


Figure 9. Second sample automata execution, demonstrating strongly duplicated pinkish cube structures. The number of duplicated structures at each iteration is displayed in Figure 8. At  $t = 421$ , three sample pinkish structures were chosen and tested for replicability.

In effect, mutations continually cull the less active controller species from the simulation, resulting in a natural selection pressure. It is worth mentioning twice that Figure 7 and Figure 8 above do not simply plot the number of cubes in the automata of a given species. Instead, they plot the total number of homogenous duplicated structures for each controller species, over time.

As was mentioned above, energy limitations on the simulated molecubes were necessary for duplication to occur (Studer and Lipson 2005). Without bounds on the maximum cubes moved per swap, the cubes tend to clump into larger, non-duplicated structures which merge themselves into even larger structures.

### Self-Replication Metric

Duplicated structures continue to emerge over long periods of time, but the existence of duplicated structures is only a prerequisite for self-replication. It is possible, for example, that some sort of non-duplicated “factory” structure is somehow creating large numbers of another structure incapable of replicating on its own. To investigate further, some way of defining and measuring self-replication needs to be applied. Choosing and justifying this measure is not a trivial task.

One definition of self-replication proposed in earlier work (Adams and Lipson 2003) is that a self-replicating structure, when placed in an environment, will create more copies of itself in the environment than would exist normally. For example, a seed crystal placed in a supersaturated solution would be self-replicating, because more crystal would form in the solution with the seed crystal than without. This idea can be quantified for a discrete system in the following way: for a replicator  $S$ , and environment  $E_0$ , self-replicability  $R$  is defined as (Adams and Lipson 2003):

$$R(S, E_S, E_0, \varepsilon, t_0, t_f) = \log \left[ \frac{\sum_{t_0}^{t_f} P_{\varepsilon}(T(E_S, t), S)}{\sum_{t_0}^{t_f} P_{\varepsilon}(T(E_0, t), S)} \right]$$

$P_{\varepsilon}$  is the “presence” function, determining how much of  $S$  is contained in some environment, and  $T$  is the time-development function for the environment, i.e.  $T(E, t) =$  the environment at time  $t$ . The  $\varepsilon$  parameter captures the tolerance to which the presence of the replicator is measured.  $E_S$  is  $E_0$  with a single replicator  $S$  inserted at some position.

Again, the intuition behind this definition is that a self-replicating unit can be detected by comparing similar systems with and without the unit. If the initial presence of this unit causes more copies of itself to emerge over time than would normally occur, then the sum of the presence of the unit over the time period would be greater for  $E_S$  than  $E_0$ , and consequently replicability would be positive. It is

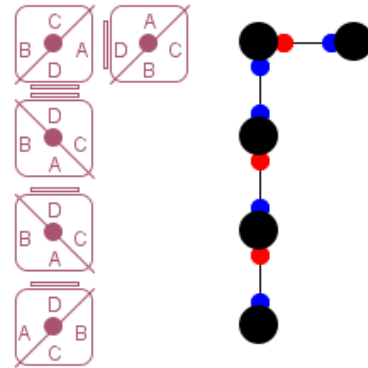
important to note that the replicability is defined over a particular period of time. It is possible for a replicating unit to have a short lifetime. As a natural example, dinosaurs had a high replicability during the Triassic, Jurassic, and Cretaceous, period when they were dominant species, according to this definition, but over the history of the planet their self-replicability drops significantly.

Because in our simulation there are so many different types of controllers, all of which are randomly generated, the chance that a particular controller will arise again over any reasonable time period is essentially zero. This is not a problem limited to our simulation, most natural self-replicating systems have this same property. For example, if measuring human replicability, the chances of a baby being born with genes identical to some other person is very close to zero too. To get around this problem, some similarity metric between structures needs to be defined. Two humans may not be identical, but they certainly are more similar to each other than to a spider, for example.

We calculate the similarity between two structures  $A, B$  as follows:

$$Similar(A, B) = SS(A, B) \times RS(A, B)$$

$SS$  (structure similarity) is the similarity of the structures only, as calculated from their magnet connections.  $A$  and  $B$ 's magnet connections are used to create graphs from  $A$  and  $B$  (Figure 10), and then these graphs are tested for isomorphism. If the graphs are isomorphic,  $SS(A, B) = 1$ , otherwise  $SS =$  (size of largest node mapping) / (size of larger structure graph). Nodes which can be mapped but contain extra connections are considered fractionally mapped.



**Figure 10. Creating a graph from a structure, which can then be used to compare structures.**

$RS$  (rule similarity) is the average similarity of the rules used in the tested structure  $B$  to the homogenous rule of structure  $A$ . While we are only checking homogenous structures for duplication, there needs to be a way to calculate how similar a homogenous structure  $A$  is to a non-homogenous structure  $B$ . Admittedly, for non-homogenous  $A$ , the *Similar* function is undefined, but we are only

applying the function to homogenous  $A$ . For each cube in the  $B$  structure with rule  $R$ , the rule similarity  $rs(R_A, R_B)$  is calculated by concatenating all the binary controller outputs from the 16 binary controller inputs for both  $R_A$  and  $R_B$ , counting up all the outputs which are the same, and then dividing by the total number of concatenated outputs.

The similarity of two structures is in this way the product of the similarity of the controllers in the structure and the topological similarity of the structures. Through the similarity measure, we can calculate the total similarity of all the structures in any automata environment to a homogenous structure, by first finding all the disjoint structures in the automata of two or more cubes, and then comparing them to the supposed replicator. This total can then be used as  $P_\epsilon$  in the replicability equation.

### Analyzing for Self-Replication

In a particular simulation instance, such as the one shown in Figure 8 and Figure 9, duplicated structures often emerge spontaneously. Shown in Figure 11 are three of the four most duplicated structures in the Figure 9 simulation at iteration 421. Note that the smaller structure (2 cubes) is part of the medium structure (3 cubes), and the large structure (5 cubes) also contains the medium and small as sub-structures. Also note that when searching the automata for disjoint structures, only *independent* copies of the structures are counted toward the structure total. So, for example, the medium sub-structure contained in the large structure would not be counted in the instances of the medium structure.

The structure duplication may be due to self-replication, but does not necessarily have to be. To determine how much these duplicated structures were actually replicating, 40 random initial automata environments were generated (with no structures initially, just initialized cubes). Each of the structures, large, medium, and small, were placed individually in these random environments at different

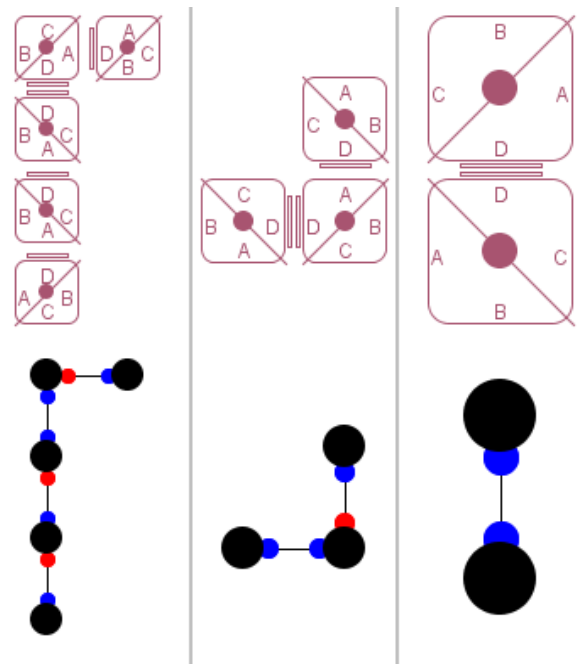


Figure 11. The three most prevalent duplicated structures in the simulation depicted in Figure 8 and Figure 9, at iteration 421. The graph representation used for the similarity comparison is shown below the structure. There were 3 copies of the large structure, 5 of the medium structure, and 12 of the small structure in the simulation at iteration 421.

random positions. The same position was used in each environment for the large, medium, and small structure. The simulation starting from each initial environment was then executed, and the presence  $P_\epsilon$  of the simulated structure (total similarity) in the environments was recorded over 4000 iterations. This value was then compared to the total similarity of the environments to the large, medium,

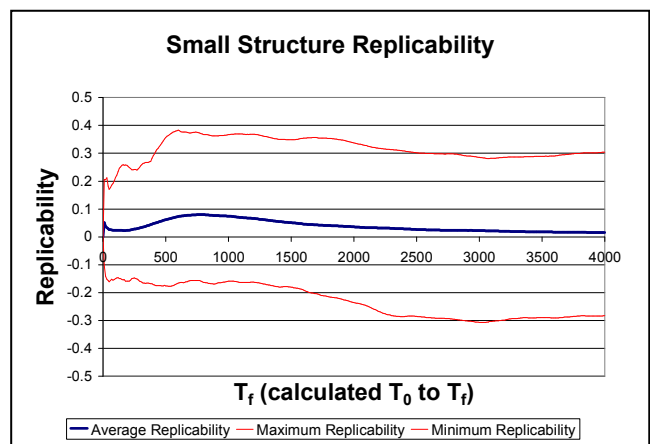
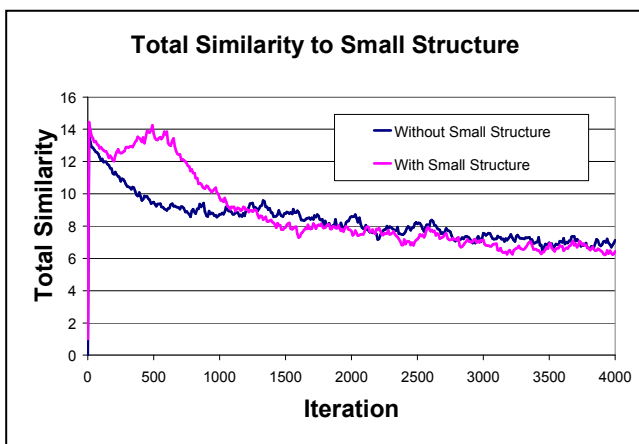


Figure 12. Average total similarity of the structures in 40 test environments to the small structure over time, with and without the initially placed small structure. The maximum, minimum, and average self-replicability of the small structure in each environment is then calculated from this data using the replicability formula.  $T_0$  is always zero when calculating replicability, and  $T_f$  is progressively increased.

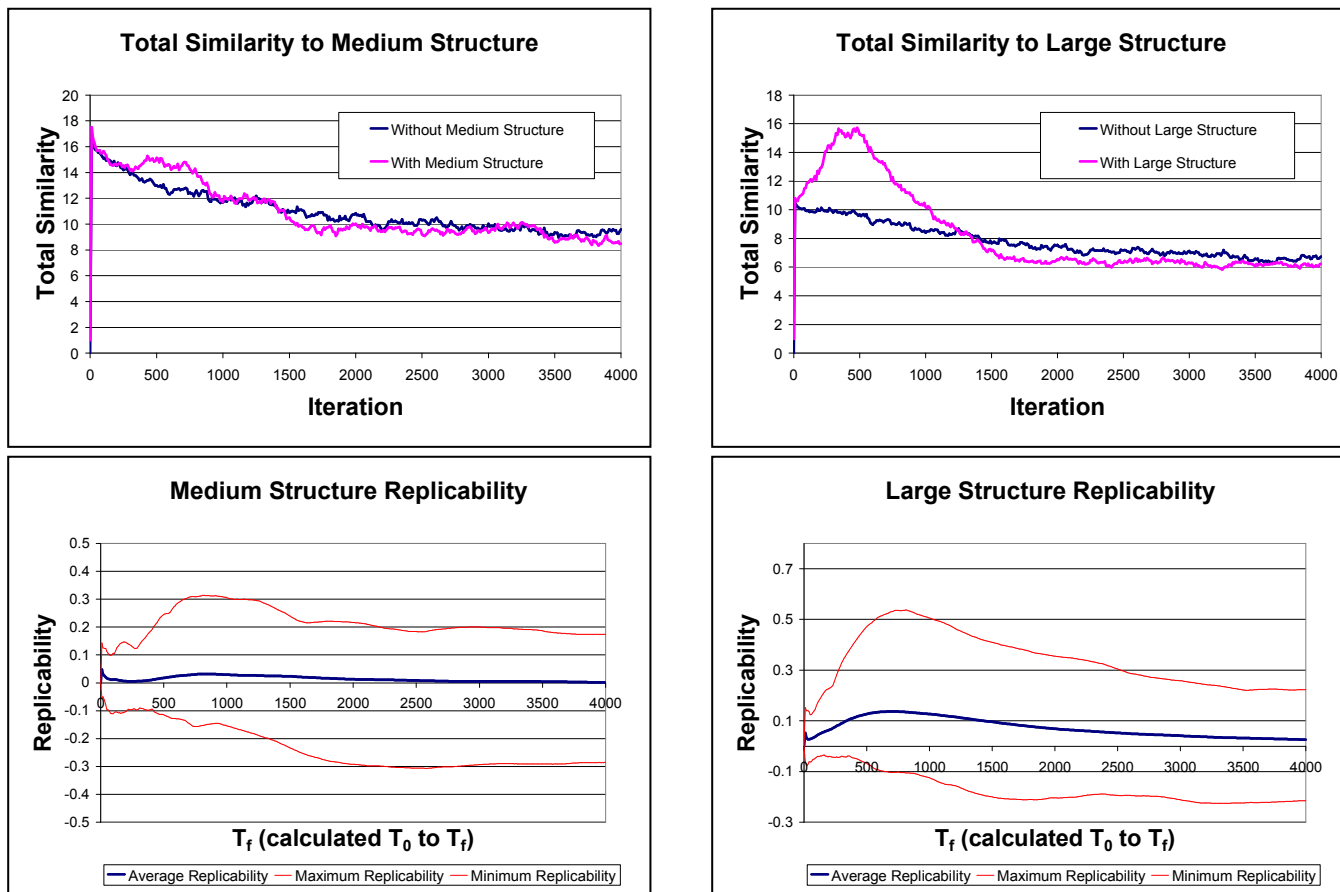


Figure 13. Total similarity of environment structures and replicability from  $0 \rightarrow T_f$  of the medium and large structures from Figure 11, evaluated in 40 random environments.

and small structures when the large, medium, and small structures were not initially added to the environments. From this data, the replicability of the large, medium, and small structures for each of the environments and over all time intervals from  $T_0 = 0$  can be calculated.

Because initial simulations are random, and there are random factors in the simulation, often the duplicated structures of Figure 11 will not be successful in the environment, and simply die out. On the other hand, they often replicate very well, and are able to use most of the other cubes to create duplicates. Averaging over many runs does not give predictable replication values, because so many different results are possible. Natural systems display this behavior as well. Bacteria require a very narrow range of temperatures and pH to grow well, so if bacteria are put in random environments with random competitors, they would tend to either die or thrive. To address this problem, maximum and minimum replicability is displayed along with the average replicability, to give error bounds for this data.

### Higher-order Interactions

As a check to see whether higher-order interactions were occurring, i.e. if different types of replicating structures were competing or cooperating over time, a correlation analysis of a sample simulation was also performed. The number of duplicated structures over time for every controller species was correlated with every other species on the time intervals where both species coexisted. In case the two populations weren't changing exactly in phase with one another, the maximum correlation over all iteration phase shifts of  $\pm 50$  iterations was calculated. The maximum correlation between populations calculated this way was found to be 0.90, not statistically significant, but analysis of more environments could be performed in the future.

### Discussion

As Figures 12 and 13 show, structures with positive self-replicability on average do spontaneously and continuously arise in the ATOMM simulation. Duplication that was

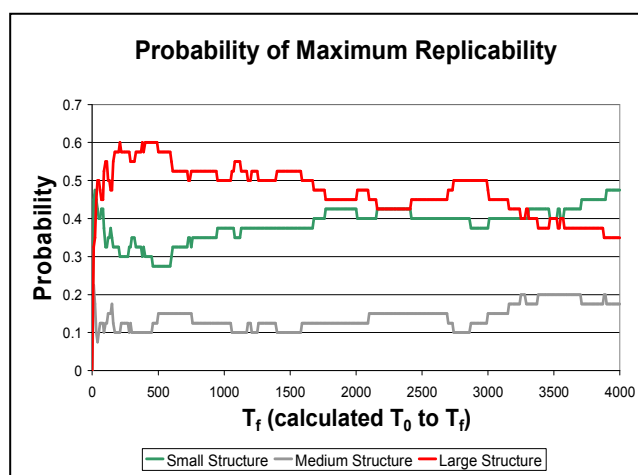
detected in the automata, shown graphically in Figure 8, was created by replicating structures. No explicit selection process is performed to evolve these replicators, only random mutations are used. Still, many replicating structures are generated over time. This implies that the space of all controllers is relatively rich in controllers that support replication, despite the limitations that no simulated molecules can be created or destroyed and limited mechanical motions of 10 cubes or less. In fact, as shown previously (Studer and Lipson 2005), replicators do not emerge at all without some limitations.

To better understand what the quantitative replicability results mean, we can compare them to another replicating system analyzed using the same metric. The best replicator observed had a maximum average replicability of 0.137 over the first 720 time steps. In a crystal growth system, where the replicability of a seed crystal in a supersaturated solution is analyzed, if there is a 0.1 probability that the seed crystal will spontaneously form at a given time in the solution, the replicability of the seed crystal is 2.35 (Adams and Lipson, 2003). If the probability the seed crystal will form spontaneously is 0.9988, the seed crystal replicability is approximately 0.137, the same average replicability as the large structures displayed in our system. The crystal growth model, however, assumes unlimited, exponential growth after a seed crystal is placed. In the ATOMM simulation, replication of a structure is severely limited by other competing cubes. The average replicability must also take those runs into account in which the structure was immediately overwritten. The maximum replicability of the structures during a particular run was much higher, reaching 0.535 for the large structure. This would correspond to a spontaneous crystal probability of about 0.75.

This comparison highlights how easy it is to produce replicating structures in the ATOMM simulated mechanical medium. At first, this may seem surprising, but given that replicators were found using only random mutation search, this must be the case. The comparison with a very unstable supersaturated chemical solution is only half the story, however. It is also possible that instead of a rich emergent environment, the replicators are simply so ineffective that they cannot generate many similar copies of themselves. This is very unlikely, as seen from Figure 9, where these small, medium, and large structures have taken over large portions of the automata. Quantitatively rating the effectiveness of a structure in an environment where the structure cannot emerge would distinguish these two cases, and is planned as future work.

### Probabilistic Replication

Because of the random nature of the simulation, individual experiments varied wildly when calculating the replicability of structures. Multiple tests from different random starting points must be performed to reliably estimate how self-replicable a structure is. The range can be large, spanning both negative and positive values. Much of this variability is due to the tested structure immediately



**Figure 14. The probability of maximum replicability is the probability that in a random environment, the replicability from  $T_0$  to  $T_f$  for a particular structure is greater than the replicability of all the other structures. This probability was calculated from the 40 tested environments.**

being overwritten before it has a chance to replicate, and often it must also compete with other replicating structures that happen to emerge naturally.

One way of avoiding this problem is to compare the relative performance of the different replicators. While individual results may vary wildly, the relative replicability of the large, medium, and small replicators when placed in the same position in the same environment may stay more constant over time. To test this hypothesis, the probability of maximum replicability in a random environment for each of the three structures was calculated. In other words, at each  $T_f$  for each of the 40 simulations, the small, medium, or large structure which had the maximum replicability was determined. By dividing the number of times the replicability of each structure was the maximum at each time step by the number of simulations, the probability that the structure was the best replicator at that  $T_f$  is determined. The results from this analysis are shown in Figure 14.

### Large vs. Medium vs. Small

As a surprising result, according to both the average replicability and probabilistic replicability measurements, more cubes are not necessarily better. The large, medium, and small structures each have different replicabilities, but greater replicability (probabilistically maximum or averaged) does not scale proportionally with structure size. It can be seen from Figure 14 that over 30% of the time, the small, two cube structure had greater replicability than the medium, three cube structure, while the medium structure only had higher replicability about 15% of the time. In addition, both the maximum average and maximum replicability for the small structure is greater than the medium structure (Figure 12 and 13). Interestingly as well, the larger structure, able to replicate more effectively than either the medium or small structure,

actually contained both the medium and small structure as sub-structures. These highly nonlinear relationships support considering multi-cube structures as individual replicating entities, more complex than simply a conglomeration of all their cube or sub-structural parts.

## Conclusions

The spontaneous and continuous emergence of small self-replicating cube structures has been observed in the ATOMM molecule simulation. The ATOMM platform could be realized using physical molecules constrained in space. The mechanical, macroscopic nature of modular robots, at least in two dimensions, does not fundamentally impede their ability to self-organize, self-assemble, and self-replicate. In addition, the controllers that make this self-assembly and self-replication possible are generally very easy to find.

Most of the duplication that occurs currently is of the smallest possible two-cube structures, but we have shown that fewer, larger duplicated structures also emerge. These smaller and larger structures both can also be capable of self-replication, though being larger does not necessarily mean that a structure is more capable. More work needs to be done to determine, in general, what makes certain replicators more effective than others. Presumably size will be a factor, but perhaps not as important as others.

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 and replicators 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 more specialized operators longer to evolve very successful replicators, but these are traits real replicating objects often use to their advantage.

Finally, it would be interesting to observe even higher levels of self-organization occurring, such as replicating structures of different types interacting. In our preliminary analysis of homogenous duplicated structures, no correlations indicating these interactions were found, but it may be that a deeper and more thorough analysis is needed to discover them. Analyzing non-homogenous structures, larger simulations, or more experiments may be necessary to observe multi-structure population behavior.

## References

- Adams B. and Lipson H. (2003). A universal framework for self-replication. *European Conference on Artificial Life*. Dortmund, Germany, pp. 1-9.
- Chirikjian, G. S., Zhou, Y., and Suthakorn, J. (2002). Self-replicating robots for lunar development. *IEEE/ASME Transactions on Mechatronics*, 7, 4, pp. 462-472.
- Chou, H. and Reggia, J. A. (1997). Emergence of self-replicating structures in a cellular automata space. *Physica D*, 110, No. 3-4, pp. 252-276.
- Freitas, R. A., Gilsbreath, W. P. (1981). A self-replicating, growing lunar factory. *Proceedings of the Fifth Princeton/AIAA Conference*, May 18-21.
- Freitas, R. A., Merkle, R. C. (2004). *Kinematic Self-Replicating Machines*. Landes Bioscience.
- Hutton, T. J. (2002). Evolvable self-replicating molecules in an artificial chemistry. *Artificial Life VIII*, pp. 341-356.
- Langdon, C. G. (1984). Self-reproduction in cellular automata. *Physica D*, 10, pp. 135-144, 1984.
- Mytilinaios, E., Desnoyer, M., Marcus, D., and Lipson, H. (2004). Designed and evolved blueprints for physical self-replicating machines. *Artificial Life IX*, pp. 15-20.
- Pargellis, A. N. (1996). The spontaneous generation of digital life. *Physica D*, 91, pp. 86-96.
- Penrose, L. S. (1959). Self-reproducing machines. *Scientific American*, 206, 6, pp. 105-114.
- Ray, T. S. (1991). An approach to the synthesis of life. *Artificial Life II*, 11, pp. 371-408.
- Sipper, M. (1995) Studying artificial life using a simple, general cellular model. *Artificial Life II*, 1, pp. 1-35.
- Smith, A., Turney, P., and Ewaschuk, R. (2003). Self-replicating machines in continuous space with virtual physics. *Artificial Life IX*, 1, pp. 405-424.
- Studer, G., and Lipson, H. (2005). Spontaneous emergence of self-replicating, competing cube species in physical cube automata. *Genetic and Evolutionary Computation Conference*, Late Breaking Paper.
- von Neumann, J. (1966). *Theory of Self-Replicating Automata*. University of Illinois Press: Urbana.
- Wilke, C. O., Wang, J. L., Ofria, C., Lenski, R. E., and Adami, C. (2001). Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature*, 412, pp. 331-333.
- Zykov, V., Mytilinaios, E., Adams, B., and Lipson, H. (2005). Self-reproducing machines. *Nature*, 435, No. 7038, pp. 163-164, 2005.