

# Self-repairing Self-assembled Structures

Daniel Arbuckle and Aristides A. G. Requicha  
Department of Computer Science, University of Southern California  
Los Angeles, California  
Email: daniel.arbuckle@usc.edu, requicha@usc.edu

**Abstract**— This paper introduces a method by which structures can be self-assembled from reactive agents, and then repaired by identical agents when the structures suffer the loss or failure of members. As a side-effect of these capabilities, the structures also have the ability to reproduce by a process analogous to mitosis.

## I. INTRODUCTION

Self-assembly agents have the potential to become an important part of the manufacturing industry. If such agents were sufficiently small, many products could be constructed by reprogramming a swarm of assembly agents.

This would replace many specialized construction processes with a single generalized one. Optimizations and improvements to this process would benefit broad swaths of products, and the process would benefit from economies of scale.

There are further benefits in that the agents could be recycled by reprogramming them for a new structure, and that damaged structures would not need to be discarded if the addition of new agents would allow them to self-repair.

The simpler the agents are, the cheaper it can be to construct them, which reduces the cost of any structure built from them. Thus it is desirable to have the agents be as computationally and physically simple as possible.

For these reasons, we present a method which allows reactive agents to construct and repair structures by joining together, using their own bodies as the building blocks of the structure.

## II. CONCEPT

Each agent is programmed with an (identical) table of message routing information. Table entries are of the form “When you receive a message matching criterion X, send message Y in direction Z.” Directions are relative, so there is no need for the agent to occupy a specific orientation or for it to have any way of knowing its orientation. A single message can trigger the sending of multiple messages in different directions.

When an agent attempts to send a message but it is not holding on to another agent in the correct direction, the agent instead attempts to grab on to an agent in that direction. In this way the rules guiding the flow of messages implicitly define the form of the structure that is constructed.

When an agent receives a message, it strengthens its hold on the agent that sent the message. Without periodic strengthening of the hold, by both involved agents, they will eventually come unbound. In this way the rules guiding the flow of messages implicitly control the removal of agents from the structure.

The structures defined in this manner may be entirely asymmetric, in contrast with many self-assembly schemes.

When the agents’ reactive tables are filled with message information, the agents gain the ability to construct and repair the shape defined by those messages. They need no further computational capacity than what is needed to execute the rules in the table, and they have no fundamental need for memory (though message buffers might be incorporated in some designs).

## III. RELATED WORK

The concepts used here were inspired by the work of our colleagues at USC in [1] and [2], then further explored by the authors in [3] and [4].

The technique described by Jones and Matarić [1] places a state machine in control of each agent; whenever the agent is joined to a new agent, or a neighboring agent changes state, a state change may occur. The current state of the machine determines the agent’s efforts to bind or unbind with neighbors. A compiler capable of automatically creating state machines for any contiguous configuration of agents in the plane was demonstrated. This system was not able to cope with agent failure or destruction, and required significant communication bandwidth between neighboring assembly agents.

In [3] and [4] Arbuckle and Requicha extend the work of Jones and Matarić, describing a family of methods based on state machine control that was i) capable of constructing non-contiguous configurations of agents, ii) configurations that had specified connectivity properties instead of a predetermined shape, and iii) known shapes around preexisting unknown shapes, thereby “repairing” them. Similarly to Jones and Matarić’s work, these systems could rarely cope with failure of assembly agents, and required significant communication bandwidth.

Klavins, Ghrist and Lipsky [5] have described robotic self-assembly in terms of graph grammars and conformational switching, resulting in systems with capability and limitations comparable to those of Jones and Matarić.

Hormone-based control as described for example in [6] and [7] partially inspired the concepts in this paper. The “hormones” presented in these papers are messages which trigger different actions in different places. The idea of explicit local communication is a powerful one for self-assembly swarms as well.

Wei-Min Shen, Peter Will and Berok Khoshnevis have applied hormone-based control to self-assembly in the context

of spaceborne operation [2], where communication between non-neighbor assembly agents is practical.

Basu, Gerchman, Collins, Arnold and Weiss have used synthetic biology to create agents that self-assemble into controlled shapes [8]. The capabilities and limitations of this technique are rather different than those discussed here.

Nagpal, Kondacs and Chang presented a biologically inspired self-assembly system with the interesting property that it relied on the ability of the agents to reproduce [9].

Stoy and Nagpal discussed a system whereby a group of agents could come together to form an object with a controlled shape but an uncontrolled size [10]. This capacity was presented as a means of self-repair and recovery from the failure of member agents, as the structure could shrink as agents failed.

#### IV. ROUTING RULES

Each agent position in the goal structure is either a “line” position or a “node” position. Node positions are at the intersection of two or more lines of agents, while line positions are the remainder of the positions.

We have developed a generic set of message rules which result in the construction of approximate lines at any angle. These rules are based on Bresenham’s integer line-drawing algorithm [11]. Agents occupying line positions refer only to these rules. When agents are reprogrammed for a new goal structure, these rules do not need to change, and so they could be implemented in hardware or read-only memory.

The message rules consulted by agents that end up in node positions are specific to each goal shape. These rules all send messages along lines such that they will eventually reach other node positions. These rules need to be changed when the agents are reprogrammed for a new shape, but remain constant throughout the construction process. They can be implemented in programmable read-only memory.

To enable self-repair, in response to receiving any message a reply message is sent back to the originating agent. This reply message contains all the information of the original, and after receipt will trigger a resending of the original message. This links each pair of adjacent agents into a message loop, each telling the other where it is in the goal structure and what messages it should send to other agents.

As an example of how messaging rules inform the construction and repair of a structure, let’s consider constructing a simple right angle between two branches. For simplicity, we won’t worry about how long the branches are supposed to be.

We start out with a pair of agents which are already bound together and exchanging messages. This is the seed from which the structure will grow. We’ll say that one of these two agents will be the joint between the two arms, though there’s no reason it has to be that way.

These two agents are exchanging messages, telling each other “you are there, because you just told me I am here” over and over. Each time one of these agents receives this message, it looks in its rules for other messages it’s supposed to send.

One of them, the one not at the joint, discovers that it’s supposed to send a message in the same direction as the

message it just received, but there’s nobody there, so it attempts to grab an agent on that side.

The other agent, the one at the joint, discovers in its rules that it should send a message at right angles to the direction of propagation of the message it just received, but there’s nobody there, so it attempts to grab an agent on that side instead.

Eventually agents are grabbed in both of the places where the seed agents are trying to grab, and each becomes part of a message loop with the agent it’s attached to. These messages cause the new agents to attempt to send messages to yet more neighbors, which causes more agents to be grabbed into the structure, and so the structure grows.

Such repetition of messages could be a significant power drain, and likewise require large bandwidth, but there are ways to optimize the problem. If the system is working correctly the messages between two agents rarely differ from one transmission to the next, so the only information that is actually transmitted by the sending of a message, in most cases, is the fact that the message was sent; a single bit of information. There are various schemes by which this fact could be used to optimize the traffic between agents without changing the basic method.

#### V. PROPERTIES OF SELF-REPAIRING SELF-ASSEMBLED STRUCTURES

If one of the agents of a pair fails, the remaining agent still has its other neighbors to tell it where it is, and so is able to continue fulfilling its role in producing the goal structure. Furthermore, if the failure is of any sort that does not cause the agent to continue sending messages in spite of its failure, and does not cause it to continue strengthening its bonds to neighbors, the agent will eventually be released from the structure, allowing a functional agent to take its place.

This self-repair capacity makes the self-assembled structure independent of its original seed agent. Any pair of operative agents which have such a message loop can re-seed the structure and cause it to regrow, and so it does not matter if the original seed (or seed pair) is destroyed, rendered incommunicado, or nonfunctional.

A potentially useful side-effect of self-repair is seen when a structure consisting of at least 4 agents is broken into two or more pieces each, containing at least one agent pair. Each piece will grow into a complete version of the structure, as long as the structures do not intergrow or otherwise interfere with each other. Much as a cell can duplicate itself by splitting, so too can self-repairing self-assembled structures.

#### VI. SIMULATIONS

The included figures are taken from simulations executing these rules. There are two sets of figures, each set taken from a single run of the simulator, and each set typical of the results obtained under similar circumstances.

Figures 1, 2 and 3 were taken from a simulation run in which all the agents within a 50-agent radius of a particular point were caused to simultaneously fail. Among the failed agents were the original seed agents of the structure. As the

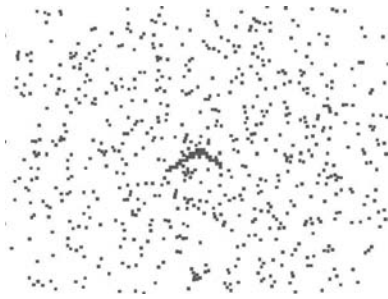


Fig. 1. Construction before localized failure



Fig. 2. Construction after localized failure

final figure in the series shows, the failed agents were ejected and the structure was correctly completed.

Figures 4, 5 and 6 were taken from a simulation run in which a partially completed structure was torn into three pieces. Each piece then grew into a complete rendition of the original goal structure.

## VII. CONCLUSION

We have presented a new method by which agents can assemble themselves into fully-specified structures, and then repair those structures when they are damaged. As a side effect of the self-repair capacity, such structures can reproduce by a process similar to mitosis, given a sufficient supply of assembly agents.

## REFERENCES

- [1] Chris V. Jones and Maja J. Mataric. "From Local to Global Behavior in Intelligent Self-Assembly". Proc. IEEE Intl. Conf. on Robotics and Automation, Taipei, Taiwan, pp. 721-726, Sep 14-19 2003
- [2] Wei-Min Shen, Peter Will, Berok Khoshnevis, "Self-Assembly in Space via Self-Reconfigurable Robots", Proc. IEEE Intl. Conf. on Robotics & Automation, Taipei, Taiwan, pp. 721-726, Sep 14-19 2003

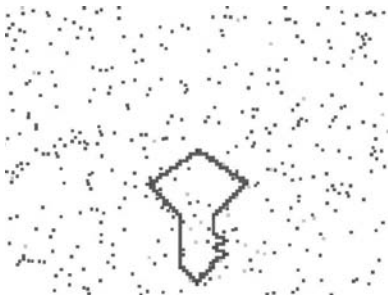


Fig. 3. Construction completed in spite of localized failure



Fig. 4. Construction before being ripped apart

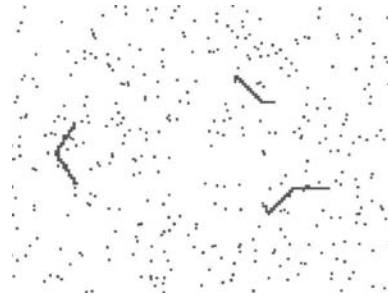


Fig. 5. Construction after being ripped apart

- [3] D. J. Arbuckle and A. A. G. Requicha, "Active self-assembly", IEEE Int'l Conf. on Robotics and Automation, New Orleans, LA, pp. 896-901, April 25-30, 2004.
- [4] D. Arbuckle and A. A. G. Requicha, "Shape restoration by active self-assembly", Int'l Symp. on Robotics and Automation, Queretaro, Mexico, August 25-27, 2004.
- [5] Eric Klavins, Robert Ghrist, David Lipsky, "Graph Grammars for Self-Assembling Robotic Systems", Proc. Intl. Conf. On Robotics and Automation, New Orleans, LA, pp. 5293-5300, Apr 26 - May 1, 2004
- [6] Shen, W.-M., Y. Lu and P. Will, "Hormone-based control for self-reconfigurable robots", Proc. Intl. Conf. Autonomous Agents, Barcelona, Spain, pp. 1-8, June 3-7 2000
- [7] B. Salemi, W.-M. Shen and P. Will, "Hormone Controlled Metamorphic Robots", Proc. IEEE Intl. Conf. on Robotics and Automation, Seoul, Korea, Vol. 4, pp. 4194-4199, May 21-26 2001
- [8] Basu, Gerchman, Collins, Arnold, and Weiss, "A Synthetic Multicellular System for Programmed Pattern Formation", Nature. April 28 2005. Vol. 434, 1130-1134.
- [9] R. Nagpal, A. Kondacs, and C. Chang. "Programming methodology for biologically inspired self-assembling systems". In Proc., AAAI Spring Symposium on Computational Synthesis: From Basic Building Blocks to High Level Functionality, 2003.
- [10] K. Stoy and R. Nagpal. "Self-Repair Through Scale Independent Self-Reconfiguration". Proceedings of IEEE/RSJ International Conference on Robots and Systems, September 28 - October 2, 2004, Sendai, Japan
- [11] J. E. Bresenham, "Algorithm for Computer Control of a Digital Plotter", IBM Systems Journal, 4:25-30, 1965

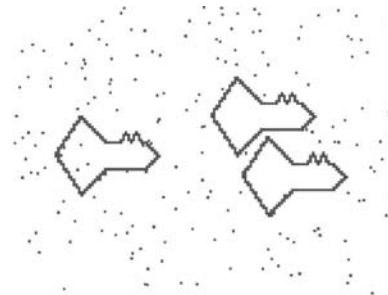


Fig. 6. Construction completed by each part after being ripped apart