On the General Reconfiguration Problem for Expanding Cube Style Modular Robots

Serguei Vassilvitskii*
Cornell University
sv39@cornell.edu

Jeremy Kubica†
Carnegie Mellon University
jkubica+@ri.cmu.edu

Eleanor Rieffel
FXPAL
rieffel@pal.xerox.com

John Suh
Xerox PARC
jwsuh@parc.xerox.com

Mark Yim
Xerox PARC
yim@parc.xerox.com

Abstract

We discuss the theoretical limitations for reconfiguration of metamorphic robots made up of Telecubes, six degree of freedom cube shaped modules, that are currently being developed at Xerox PARC. We show that by using meta-modules composed of 8 individual modules as a backbone for building the desired shape, we can establish completeness for the reconfiguration as well as time and space bounds for the process. Finally we present several open problems in the field of reconfiguration.

1 Introduction

We are developing a versatile modular self-reconfigurable robotic system consisting of identical units that are very limited in their actions. However, as the number of modules increases, the range of behaviors of the system grows exponentially. The task of self-reconfiguration is important for developing self-sufficient systems. The overall system can reconfigure itself to help accomplish certain tasks such as locomotion, object manipulation and sorting, or interaction with other systems, especially when there is a need to adapt to the environment.

Many people have realized that by grouping single modules into groups or meta-modules each unit in the system increases its number of degrees of freedom and the reconfiguration tasks are simplified [3, 4, 6, 9]. However using meta-modules limits the granularity of the possible configurations. Rus and Vona require 4x4 meta-modules for complete 2D reconfiguration for expanding cube style modules, while Nguyen et al. explore the possibility of 36 membered meta-modules for 2D reconfiguration with hexagonal modules. We explore the use of small meta-modules as the backbone for the entire structure, but allow other modules to be embedded within the meta-module structure. This approach results in a smaller loss of fine structure, yet still retains the advantages of the meta-module method.

We first describe the hardware platform currently being developed at Xerox PARC that inspires our work. In section 3 we motivate the need for meta-modules by demonstrating the existence of structures that cannot reconfigure into other structures. We then describe the use of 2x2x2 meta-modules to aid in reconfiguration, and demonstrate that reconfiguration can be performed in-place in quadratic time. We deal with the 3D reconfiguration problem explicitly, but note that the same proofs carry over to using 2x2 meta-modules for 2D reconfiguration. Finally, we present several open problems that we are currently working on.

2 Telecube Modular Robotic Hardware

We look at the Telecube generation of modular robots designed and currently being constructed at Xerox PARC. Similar in design to the Crystalline modules used by Rus [6, 8], these modules are cube shaped. The Telecube modules have the ability to independently extend out each of the 6 faces. These extensions and retractions provide the modules' only form of motion. A picture of a module is shown in Figure 1.

The arms can extend independently up to half of the body length, giving the robot an overall 2:1 expansion ratio along each dimension. A latching mech-

*Supported by Xerox PARC
†Supported by FXPAL

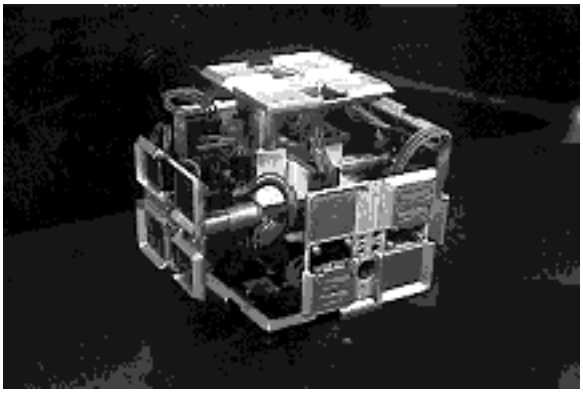

Figure 1: One telecube module.

anism on the plates on the end of each arm enables two aligned modules to connect to each other. For power routing, communication and alignment reasons, the modules must remain globally connected to each other at all times. While the modules are in development, we have built a simulator to develop and test reconfiguration algorithms.

The modules have the following low-level primitives:

- *ExtendArm(Direction)*: If there is room to extend the arm in *Direction*, extend the arm.
- *RetractArm(Direction)*: Retract the arm in *Direction*, attempting to first disconnect from neighbor if connected.
- *Connect(Direction)*: If there is a neighboring module in *Direction*, latch to that module.
- *Disconnect(Direction)*: If there is a module is currently latched in *Direction*, break the connection with the neighbor.

From these primitives, we can build more complicated actions, such as *Move(Direction)*. The explicit sequence of actions that allows a module to move along a given direction is illustrated in Figure 2. A module can pull towards a neighbor by retracting its arm, push away from a neighbor by expanding its arm, or simultaneously retract its front arm and expand its back arm, effectively "sliding along its arms" in a given direction. Prior to moving, the module:

1. confirms that it has at least one neighbor along the direction of motion on which it can push or pull,
2. ascertains that it is moving into free space,
3. disconnects from all neighbors perpendicular to the direction of movement

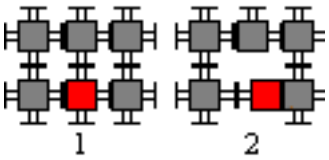

Figure 2: The module in red moves one arm length

At any point during this process, the movement can fail, in which case the module reverses what has been done so far and returns to its original state.

Each module is also given simple sensing and communication abilities. Modules can send messages through their faceplates to their immediate neighbors using a low bandwidth IR link. Each module can also gauge the extension of each faceplate, read the contact sensor on each of the faces, and determine whether it is latched to a neighboring module.

3 Related Work

The problem of reconfiguration for modular self-reconfigurable robotic systems has received increased interest. This work includes [2, 3, 5, 6, 7, 10, 11, 12, 13]. Much of the work on the theoretical bounds of reconfiguration has centered on two dimensional hexagonal modules that move by rolling over other modules. In [4] and [5], Pamecha and Chirikjian examine the theoretic bounds of reconfiguration on such a system, including the upper and lower bounds on the minimum number of moves required for reconfiguration. Further, they propose the use of simulated annealing to attempt to solve the reconfiguration problem. Walter, et. al. describe distributed algorithms for such a system and prove correctness and lower bounds for reconfiguration in [12] and [13]. Rus and Vona describe a general reconfiguration algorithm for cube modules in [6]. Their melt-grow algorithm uses blocks composed of 16 modules in a 4x4 arrangement and is shown to be complete for structures composed of such blocks. Little has been done, however, to explicitly tackle the problem for 3D reconfiguration.

Previous work on reconfiguration often encounters inadmissible configurations [3, 13] or other constraints [6]. We further classify different configurations as static or dynamic to give more motivation for the need of meta- modules. Finally, we address the problem of bounding the physical space used during reconfiguration. This bound is of crucial importance for situations in which the robots need to reconfigure in tight spaces or in the presence of obstacles.

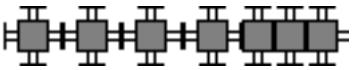

Figure 3: Module arranged in a line cannot reconfigure into a different shape

4 Motivation

Ideally, we would like to have a reconfiguration algorithm that allows any arbitrary structure of modules to reconfigure into any other arbitrary structure. However, this is not feasible in general because there exist pairs of structures between which reconfiguration is not possible. For example, modules arranged in a straight line, as shown in Figure 3, cannot reconfigure into any other structure. This simple fact demonstrates the inherent difficulty of the reconfiguration problem and shows the need to first determine if reconfiguration is possible between two given shapes.

Walter et. al. and Nguyen et. al. approach this problem by defining various inadmissible configurations for the hexagonal modules [3, 13]. This serves as a useful heuristic, but includes configurations where reconfiguration is still possible. Using the Telecube type module it is possible to precisely define a set of configurations that are rigid in that not only can they not reconfigure into a different shape, they cannot even change any of their connections. Thus, we can partition all possible configurations into two disjoint subsets: rigid and dynamic.

4.1 Definitions

We start with the following definitions:
Structure: Any connected set of 2 or more modules.
Connectivity Arrangement: The set of connections between modules in a structure.
Rigid structure: Any structure that cannot change its connectivity arrangement while maintaining global connectivity.
Dynamic Structure: Any structure that can change its connectivity arrangement while maintaining global connectivity.

We will use n to refer to the number of modules in a structure.

4.2 Constructions

Theorem 1: A line of connected modules is a rigid structure.
Proof: In order to change its connectivity arrangement, a module has to either make a new connection or break a connection. Since the modules are connected and arranged in a line, all possible connections between modules are already made. Further, breaking any connection in the line would break the global connectivity constraint.

Theorem 2: If any two rigid structures, s_1 and s_2, establish one connection, and cannot establish other connections regardless of further extensions and/or retractions of their faceplates, then the resulting structure is rigid.
Proof: Let s be a structure formed by establishing one connection between two rigid structures, s_1 and s_2. As no new connections can be formed within a rigid structure and one of the assumptions of the theorem is that no other connections can be formed between structures s_1 and s_2, we need only show that no existing connections can be broken. The single connection between s_1 and s_2 cannot be broken, since no other connections between the two structures are possible. Therefore, severing this connection disconnects structures s_1 and s_2 violating the global connectivity constraint. Any connection within s_1, disconnects s_1 into two pieces. As there is only one connection with s_2, only one of these pieces will be connected with s_2, so the resulting structure is disconnected. The argument for any connection within s_2 is similar.

Theorem 3: If any two rigid structures, s_1 and s_2, establish a connection and can establish further connections between the two structures, the resulting structure is dynamic.
Proof: Let p_1 and p_2 be two of the connection points. Note that if there is a connection at p_1 then the module(s) at p_2 are free to make or break that connection while maintaining global connectivity. Thus the resulting structure can change its connectivity arrangement and is no longer rigid.

Theorem 4: If any dynamic structure establishes a connection with any other structure, the resulting structure is dynamic.
Proof: Let M be a module in one of the two structures that could change the connectivity arrangement by making or breaking a connection to a neighboring block, M', without causing a global disconnect in the original structure. Thus there exists another path from M to M' in the original structure that did not go through this connection. Since this property still holds in the resulting structure, the resulting structure is dynamic.

Note that there exist dynamic structures that can

change their connectivity arrangements, but cannot change their shape. For example, a 2x2 completely connected square of modules is dynamic because any one of its connections can be broken without violating the global connectivity constraint, but the modules cannot change their position relative to their neighbors.

The existence of rigid structures shows that reconfiguration is not always possible. Classifying the dynamic structures into disjoint classes of mutually reconfigurable structures is an interesting open question. However, if we group the modules into the smallest symmetrical group - 8 modules arranged in a 2x2x2 cubic configuration - the question of which structures can reconfigure into other structures becomes easy.

5 The Use of Meta-Modules with Free Blocks

The technique of combining individual modules into meta-modules has proven beneficial in simplifying the problem of reconfiguration, easing the global connectivity concern and eliminating the problems of inadmissible states [3, 6]. However, the meta-module technique has several drawbacks:

- The precision of shape approximation is now limited by the size of the meta-module.
- Reconfiguration using meta-modules may not be optimal, because the meta-modules need to be kept intact at the end of each step of the reconfiguration.
- In cases where the modules are interacting with other objects, the amount of fine control with which the modules can interact may again be limited by the size of the meta-module.

All of the above downsides of the meta-module approach are strictly related to the size of the meta-module. Thus, it is useful to find the smallest meta-modules that can still simplify the reconfiguration problem.

We extend on the previously explored idea of meta-modules [3, 6, 4] by allowing free blocks to be part of the overall structure. By allowing free blocks within the meta-module structures, we greatly expand the set of possible shapes and also increase the precision with which an arbitrary shape can be approximated. This meta-module approach eliminates inadmissible states and allows for arbitrary reconfigurations within this set of shapes. Further, we show that this approach to the meta-modules can be used to make guarantees about the space required by the structure during the reconfiguration. Specifically, we show that the space required by the modules for reconfiguration can be bounded by the space required by the initial and final structures. This guarantee is important when the robots have to reconfigure in tight spaces, around obstacles, or if two systems of modules are reconfiguring in close proximity to each other.

6 Results

Below we prove that reconfiguration using meta-modules with free blocks is always possible between two systems with the same number of modules n and, furthermore can always be done in place, with no additional space, in $O(n^2)$ total module movements.

6.1 Definitions

Lattice Node: A meta-module composed of 8 modules arranged in a 3 dimensional cube with 2 modules on each side, spaced apart by two full arm lengths. (Figure 4a).
Node Module: Any module currently part of a lattice node.
Valid Space: A space between any two node modules not necessarily of the same node.
Free Block: A module occupying a valid space. (Figure 4b). Note that at any given time, the set of modules that are acting as node modules and the set of free blocks are completely disjoint.

Lattice Structure: Any connected structure of aligned lattice nodes and free blocks. A lattice node A is said to be aligned with the neighboring lattice node B if and only if all of the modules of A facing B are connected to modules in B and vice versa.

We will take N to be the number of nodes. Note that $O(N) = O(n)$, because $n \leq 20N$.

Figure 4: a) Modules and (b) A free block in a 2x2 meta-module configuration.

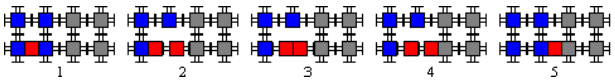

Figure 5: Free block moving to an adjacent valid space (only one layer of the meta- module shown).

We will make frequent use of the homogeneity of the modules in order to accomplish reconfiguration tasks. In particular, modules that correspond to free blocks in the original structure may correspond to lattice nodes in the final structure, and vice versa.

We first describe two basic moves that will be generally useful for reconfiguration.

Translation: Any free block can move to an empty valid space on the opposite side of a neighboring node module. Further, this can always be done while maintaining global connectivity.

The exact sequence of moves for one case is demonstrated in Figure 5. Global connectivity is maintained at all times. Note that a module that initially corresponded to a free block became a node module and vice versa.

Turn: Any free block can move to any other empty valid location adjacent to a neighboring node module. Further, this can always be done while maintaining global connectivity.

Using translation, a free block can move to the other side of a node module. The block can also "turn": move in the four directions perpendicular to the direction of the module's connection. The exact sequence of moves is demonstrated in Figure 6. Again, the global connectivity constraint is never violated.

6.2 Constructions

Theorem 5: Any free block can move to any empty valid space within a lattice structure while maintaining global connectivity. Moreover, this can be done in $O(N)$ time, where N is the number of lattice nodes.
Proof: Since all of lattice nodes in a lattice structure are connected, there exists a path from any lattice node to any other lattice node in the structure. Moreover, there is a path between any two valid spaces

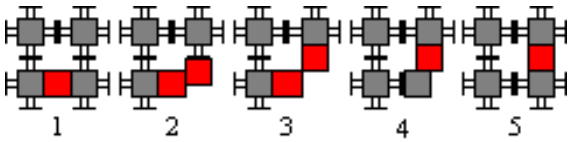

Figure 6: Free module moving to an adjacent valid space (only one layer of the meta- module shown).

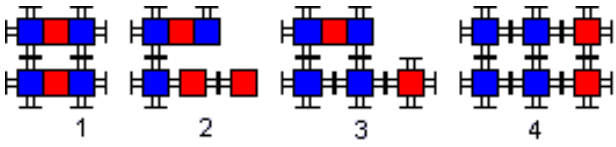

Figure 7: Two free blocks pushing out to form the basis of a new node (only one layer of the meta-module shown).

in the structure consisting of a sequence of adjacent valid space. Any such path can be broken up into a series of translations and turns, which a free block can execute using the primitive motions defined above. Since it takes at most 3 turns for a free block to move from any valid space within a lattice node to any other, and two translations to move between nodes, the length of this path is at most $5 * N$.

Finally we look at the case where the path a free block must take already contains free blocks. We examine each of the free blocks encountered, F_1, F_2, F_3, ..., F_m. We move the final free block, F_m to the final location. We then iterate by moving F_{m-1} to the location previously occupied by F_m, F_{m-2} to the location previous occupied by F_{m-1}, etc. Note that since the blocks are moved in the reverse order, none of them are blocked. The end result is that the original free block has effectively moved through the other free blocks, though no single module has followed the complete path.

Theorem 6: Eight free blocks can be used to form a new lattice node in any empty location adjacent to and aligned with the lattice structure. Further, this can always be done in O(N) time while maintaining global connectivity.
Proof: By Theorem 5, we let 4 of the free blocks occupy the valid spaces inside a node adjacent to the new location. Then, by the sequence of steps shown in Figure 7, these 4 free blocks can be pushed out to form half of a lattice node while maintaining global connectivity.

We can now move the remaining 4 free blocks to the spaces between the node and the half node, and repeat the above procedure to push out another half node. Since the half node is connected to the lattice structure at four points, global connectivity is preserved throughout the movement of the free blocks and the formation of the rest of the node.

Corollary 2: A lattice node can break apart into 8 free blocks while maintaining global connectivity if there are 8 empty valid spaces in the rest of the lattice structure and the removal of the lattice node

would not cause a global disconnect.
Proof: The sequence of steps above can be reversed to break up a lattice node into 8 free blocks occupying the appropriate valid spaces.

Theorem 7: A lattice node containing no free blocks can move from any position in a lattice structure to any other position adjacent to and aligned with any other node in the lattice structure while maintaining global connectivity as long as the global connectivity of the lattice structure would be maintained without the presence of the node. Moreover, this movement has a runtime of O(N), and requires no extra physical space, other than that occupied by the initial and final structures.
Proof: The overall approach for the proof is that by following Corollary 2 and Theorems 5 and 6, a lattice node can break up into 8 free blocks that can migrate to the final position and build a new lattice node. There are three cases to consider:

Case 1: The lattice has at least 8 free blocks. By Theorem 5 and 6, eight of these free blocks can move and form a new node in the final position of the moving node. By Corollary 2 and Theorem 5, the eight modules in the moving node can break down into eight free blocks and move to the positions previously occupied by the eight free blocks that formed the new node.

Case 2: The lattice has fewer than 5 free blocks. This means that the lattice has at least 8 empty valid spaces, because a single node has 12 valid spaces. Using Corollary 2, the node can break up into 8 free blocks, reducing to Case 1.

Case 3: The lattice has 5 to 7 free blocks. As the proofs for Theorems 5 and 6 show, four of the free blocks can move to the new position and form half of the new lattice node. As shown in case 2, there are then at least 8 empty valid spaces in the structure. By Corollary 2, the moving node can break into 8 blocks, 4 of which can move to move and complete the new node and four of which can move into the spaces previously occupied by the 4 free blocks that formed the first half of the new node.

The process of building a node and breaking up a node as described in Theorem 6, is constant in time and is independent of the total number of lattice nodes. The migration of a free block from one lattice nodes to another has a runtime of O(N) as demonstrated in Theorem 5. Therefore, this movement of a lattice node can be accomplished in linear time. Furthermore, since after breaking up into free blocks the lattice node "tunnels" through the initial structure, this process does not require any additional space.

Theorem 8: Any lattice structure L_i can reconfigure into any other lattice structure L_f with an equal number of modules, where the two overlap by at least one lattice node, while maintaining global connectivity. This reconfiguration can be done in $O(N^2)$ time, and requires only the space of the initial and final configurations.

First, we define S_i to be the set of module positions in the initial structure and S_f to be the set of module positions in the final structure. Choose an arbitrary anchor node A such that $A \in S_i$ and $A \in S_f$. Such a node is guaranteed to exist by the assumption that the initial and final structures overlap by at least one node. Any nodes contained in both S_i and S_f that have a path to A that remains in both S_i and S_f do not need to move, so we define the set of in-place nodes, I, as the maximal connected subset of nodes such that all nodes in I have paths to A that are entirely contained in both S_i and S_f.

In addition, we assign all nodes B in S_i a distance value d(B) from I. The distance value d(B) is the number of nodes contained in a shortest path from B to some node in I, where the path must lie completely inside the lattice structure. Thus, any node in I has a distance value of zero and any node neighboring a node in I has distance value one. We also assign distance values to all node locations (currently empty) in the final structure using S_f's connection structure.

Lemma 1: If a node H in S_i with the highest distance value is removed from the initial structure, the resulting structure will remain connected.
Proof: Assume the shortest path to I for some node B went through node H. By the definition of path distance, B's path distance would then be strictly greater than H's, which violates the assumption that H has the highest distance value. Thus every node has a path to I that does not go through node H, so removing H does not disconnect the structure.

Lemma 2: The number of free blocks in the initial and final structure can only differ by a multiple of 8.
Proof: The number of free blocks can be described by the following formula: $F = n - N * 8$.
Since the total number of modules is unchanged between the two structures, and the number of

lattice nodes is an integer, the total number of free blocks can only differ by a multiple of 8.

Lemma 3: If there is an empty node in S_f, there is one with distance value 1.
Proof: Let P be an empty node in S_f. Let B be the final node in the shortest path from P to I. Node B must have distance value 1.

Proof of Theorem 8:
Step 1: If all nodes in S_f are filled, skip to Step 2. Otherwise, by Lemma 3 there is an empty node E in S_f next to I. If there exist 8 free blocks in the lattice structure L_i, these can be used to fill the empty node E by Theorem 6. If there are fewer than 8 free blocks, there must be a node in S_i with distance value greater than 0 since otherwise L_i would have fewer modules than L_f. Let H be a node in S_i with highest distance value. If it contains any free blocks, move them into I using Theorem 5. Then use Theorem 7 to move the modules in H to build node E. Let the resulting structure be the new lattice structure L_i, and repeat this step until all of the nodes in S_f have been filled.

Step 2: If all of the nodes in S_f are filled, we only need to place the free blocks in the right places. If the number of nodes in S_f is smaller than the number of nodes in S_i, then the remaining nodes in S_i that are not part of S_f can be broken down into free modules, always choosing the node with the highest distance value. By Lemma 1 and Corollary 2 this will not cause a global disconnection. We now have the same number of free blocks as in the final configuration. These free blocks can migrate to the correct locations by Theorem 5.

After one iteration of Step 1, the set I has increased by at least one element. Thus, the complete construction of S_f can be done in at most N iterations of Step 1. Since each iteration of Step 1 also requires no more than O(N) time by Theorems 5, 6 and 7, the complete construction of S_f can be achieved in $O(N^2)$ time. The number of free blocks is also of O(N), and moving any one of them to the desired position again requires only O(N) time by Theorem 5, so Step 2 also can be achieved in $O(N^2)$ time. Due to the fact that each movement does not require additional space, the complete reconfiguration does not require any additional space.

7 Discussion

Above we show that meta-modules can be used to define a class of structures in which reconfiguration is complete and efficient both in time and space. The meta-modules themselves present an underlying structure to the lattice that facilitates the reconfiguration, and free blocks can be added to the structures at no penalty to the bounds proved above. This lack of restriction provides significant power over a pure meta-module approach that has been previously suggested in [3, 6, 9]. For example, the free blocks can be placed at valid locations along a single edge of the structures surface to provide a relatively solid wall. By moving the free blocks within the meta-module structure, it is possible to manipulate the density at any particular point. Finally, free blocks can be used in self-repair by replacing defective node modules.

Based on the proof for Theorem 8, we can implement a reconfiguration algorithm where one meta-module would move through the lattice structure at a time. Similar to the method introduced by Pamecha and Chirikjian for hexagonal modules [5] we can prove a worst case lower bound of $O(N^2)$ moves on arbitrary configurations: consider, for example, a horizontal line of meta-modules reconfiguring into a vertical line. This, however, may not be a tight bound for concurrent reconfiguration, in which more than one module can move simultaneously.

It is important to note that although we can bound the number of moves required for reconfiguration, the problem of planning an optimal reconfiguration is non-trivial. The search space is highly exponential: $O(4^n)$ for 2-d structures and $O(8^n)$ for 3-d structures [1]. To the best of our knowledge there does not exist an algorithm to solve the planning problem in polynomial time. The reason for this, as mentioned by Pamecha and Chirikjian, is its relation with the shortest path problem on a graph with an exponential number of nodes.

Though not of great practical interest given the generality of the classes we defined, it would be nice to have a complete classification of structures into reconfigurations classes. Furthermore, there are related reconfiguration problems of interest. For instance, if the degrees of freedom of the modules were reduced, say the opposite arms had to expand or contract at the same time, or even that all the arms contract or expand together,what can be said about reconfiguration under these constraints?

8 Conclusions

The general shape reconfiguration is not possible for a large class of structures due to their rigidity. However, we have shown that by grouping single modules into groups of 8, and allowing additional single modules within the structures formed by the groups of eight modules, the problem becomes much more feasible and can be solved in the general case. We further note that the reconfiguration done with the use of meta-modules can be performed in place, and can thus be carried out in tight spaces. Thus, by defining this class of structures we are able to make guarantees about the possibility of reconfiguration and the time and space required to execute it.

References

[1] Finch, Steven. Klarner's Lattice Animal Constant. 16 Sept. 2001 http://www.mathsoft.com/asolve/constant/rndprc/anml.html

[2] Kotay, K. Rus, D.: Motion Synthesis for the Self-reconfiguring Molecule. Proceedings of the 1998 International Conference on Intelligent Robots and Systems.

[3] Nguyen, A., Guibas, L., Yim, M.: Controlled Module Density Helps Reconfiguration Planning. New Directions in Algorithmic and Computational Robotics. A. K. Peters 2001. 23 - 36.

[4] Pamecha, A. and Chirikjian, G.: A Useful Metric for Modular Robot Motion Planning. JHU Technical Report, RMS-9-95-1.

[5] Pamecha, A. and Chirikjian, G.: A Bounds for Self- Reconfiguration of Metamorphic Robots. JHU Technical Report, RMS-9-95-2.

[6] Rus, D., Vona, M.: Crystalline Robots: Self-reconfiguration with Compressible Unit Modules. Autonomous Robots. January 2001. 10 (1): 107-124.

[7] Rus, D., Vona, M.: Self-Reconfiguration Planning with Compressible Unit Modules. Proceedings of the 1999 IEEE Int. Conference on Robotics and Automation. 2513-2520.

[8] Rus, D. Vona, M. A Physical Implementation of the Crystalline Robot. Proceedings of the 2000 IEEE Int. Conference of Robotics and Automation.

[9] McGray, C., Rus, D.: Self-Reconfigurable Molecule Robots as 3D Metamorphic Robots. Proceedings of the 1998 Conference on Intelligent Robot Systems.

[10] Unsal, C., Khosla, P.: Solutions for 3-D Self-reconfiguration in a Modular Robotic System: Implementation and Motion Planning. Proceedings of SPIE, Sensor Fusion and Decentralized Control in Robotic Systems III, SPIE, Vol. 4196, November, 2000.

[11] Unsal, C., Kiliccote, H., Patton, M., Khosla, P.: Motion Planning for a Modular Self-Reconfiguring Robotic System. Distributed Autonomous Robotic Systems 4, Springer, November, 2000.

[12] Walter, J., Welch, J. and Amato, N.: Distributed Reconfiguration of Metamorphic Robot Chains. Proceedings of the Nineteenth Annual ACM SIGACT- SIGOPS Symposium on Principles of Distributed Computing (PODC 2000), Portland, Oregon, 2000, pp. 171-180.

[13] Walter, J., Welch, J. and Amato, N.: Distributed Reconfiguration of Hexagonal Metamorphic Robots in Two Dimensions. Sensor Fusion and Decentralized Control in Robotic Systems III, Gerard T. McKee and Paul S. Schenker, eds., Proceedings of SPIE, Vol. 4196, pp. 441-453, 2000.