

Cellular Automata for Decentralized Control of Self-Reconfigurable Robots

Zack Butler*, Keith Kotay*, Daniela Rus* and Kohji Tomita*†

*Department of Computer Science

Dartmouth College
Hanover, NH, USA

†National Institute of

Advanced Industrial Science & Technology
Tsukuba, Japan

Introduction The key research questions in self-reconfiguring robotics are how to design and engineer robot systems capable of self-reconfiguration, and how to plan for such systems. Several interesting robot designs have been proposed [RV01, KR99, MKY⁺98, TMK⁺99, FK90, PCSC96, YDR00, SWC99, UK00]. For each of these systems, architecture-dependent algorithms that couple planning to the specific actuation have been proposed. This includes real breakthroughs in both centralized planning [KR99, YMK⁺00] and decentralized planning [BBR, MYK⁺01, TMK⁺99]. This body of work has brought some real insight into planning and control.

We believe we are at a point where we can step back and examine more general questions about self-reconfiguration planning in an architecture-independent way. It is important to examine architecture-independent algorithms that can be instantiated to many different systems because they have the potential of providing a more general science-base for the field. By outlining general principles for reconfiguration planning, we hope to learn how to better design hardware and control algorithms.

In most existing self-reconfiguring robot systems, an individual module can move in general ways relative to a structure of modules, by traveling on the surface of the structure. By composing such movements, it is possible to generate a planner for shape transformation or for locomotion. The details for how to accomplish these goals are architecture-dependent. Automating this process, which is necessary to manage the individual degrees of freedom in robots made of many modules, has been the theme of the previous work in self-reconfiguration planning. We observe that we can abstract out three types of movement for the motion of one module on the surface of a structure made of lots of modules: (1) linear motion on plane of modules; (2) convex transitions into a different plane; and (3) concave transitions into a different plane. We use these abstractions as the basis of a general decentralized algorithm that can work for any system capable of such motion. In this paper we describe a distributed control algorithm for locomotion based on this abstract model.

Motivation for cellular automata The ability of a self-reconfiguring system to change shape can lead to “water-flow”-like algorithms that can conform to the terrain on which the robot has to travel and have the potential of working well in unstructured environments. To create “water-flow”-like locomotion, we use a cellular automata model. We assume that modules on the surface of the robot can travel in arbitrary way on the surface, as long as they do not disconnect the structure. We represent the basic unit by cubes, but this is just the abstraction used for the algorithm. We describe instantiations of this algorithm to specific systems where the basic unit is not a cube.

Locomotion without obstacles The simplest locomotion algorithm is one without explicit obstacles (we consider the floor to be composed of implicit obstacles). The rules shown in Fig. 1 are sufficient to move a rectangular array of cells eastward, where eastward motion is to the right (equivalent to the positive x axis, north is equivalent to the positive y axis). The rules are shown as productions, in which the left side represents the preconditions which must exist in the environment of the cell being processed, the “current cell”, represented in this figure by a black square with a white dot. The right side of the production represents the environment of the current cell after the rule has been applied. The rules in Fig. 1 require three different existence tests: whether a cell exists at a

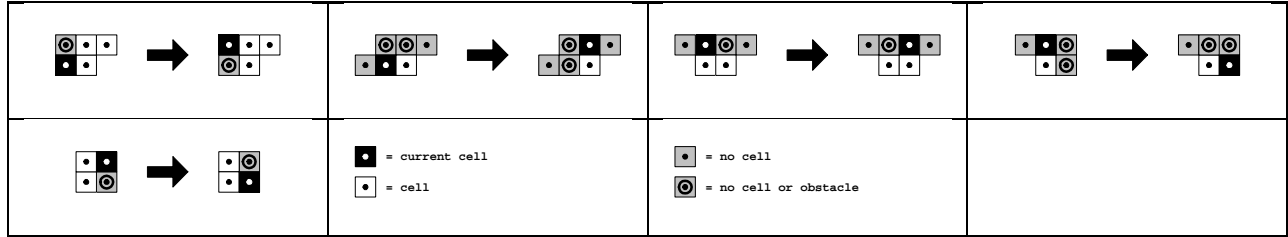


Figure 1: Five rules for eastward locomotion without obstacles.

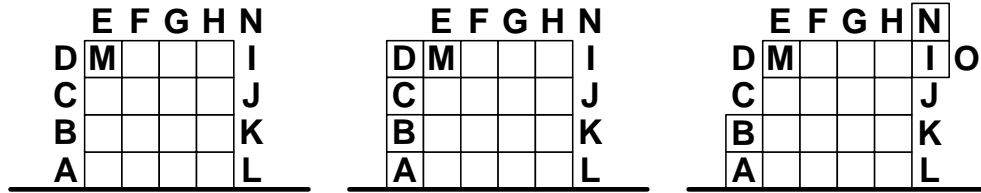


Figure 2: Diagrams for the proofs of Lemmas 1-3.

location, whether a cell does not exist at a location, and whether a cell is empty. For example, Rule 5 in Fig. 1 can be applied only if there is an empty location below the current cell, and existing cells to the west and southwest. If Rule 5 is applied, then the result will be that the current cell has moved by one in the southward direction.

It is important to be able to prove that any rule set produces the intended result. In order to show that the rule set in Fig. 1 produces eastward locomotion on a rectangular array of cells, it is sufficient to show that 1) some rule from the rule set can always be applied to the cell array, 2) eastward motion must result from all possible sequences of rule activations, and 3) the array remains connected, i.e. the cell array always consists of one simply connected component. These claims are proven in the following three lemmas. The lemmas assume that the cell array meets some minimum width and height bounds which do not preclude any rule from being applied. In the case of the rule set in Fig. 1 the minimum width is three cells and the minimum height is two cells.

Lemma 1 and Lemma 3 also assume that the “random round-robin” simulation method is used. Random round-robin means that cells are processed in random order within a “round,” but that all cells are processed in each round. This method produces a bound on the relative activation frequency of any two cells: a cell can be processed at most twice between two processing times of some other cell. For example, cell P could be processed before cell Q in round 1, and cell Q could be processed before cell P in round 2. The result is that Q was processed twice between the processing times of P . This type of pseudo-synchronous processing seems reasonable, given that in real systems communication is much faster than actuation, so cells would quickly decide whether they should move as soon as their environment allowed a rule actuation, i.e. they would move in a timely manner. It is possible to develop rule sets that are asynchronous, thereby allowing arbitrary delays in actuation, but the rule set tends to be more complex.

Lemma 1 *Given a rectangular cell array, in the absence of disconnections some rule from the rule set shown in Fig. 1 can always be applied.*

Proof: The general idea of how this rule set works is that cells along the westward edge of the cell array move to the top of cell array, where they then move eastward until reaching the eastern edge of the cell array. At that point, the cells move southward until no more motion rules apply. Thus, the motion is akin to that of a caterpillar tread.

Considering the left diagram of Fig. 2, we see a generic cell array with border locations labeled $A-L$ (for convenience cells are indicated without the black dot in the center). It is easy to show that if a cell X is located in any of the locations $A-L$ and the rules in Fig. 1 are continuously applied only to X , then X will eventually come to rest in location L . This is because at any location only one rule can be applied: at locations $A-C$ only

Rule 1 applies and results in northward movement, at location D only Rule 2 applies and results in northeastward movement, at locations $E-G$ only Rule 3 applies and results in eastward movement, at location H only Rule 4 applies and results in southeastward movement, at locations $I-K$ only Rule 5 applies and results in southward movement, and at location L no rule applies. Note that increasing the height or width of the array would only add more locations where a specific rule applies, it would not change the number of rules that could be applied at any location.

Furthermore, by examination of the rule set it can be seen that placing a cell, X , in any of the locations $A-L$ cannot satisfy the preconditions for a rule to be applied to any other cell.

Now consider the diagram in the center of Fig. 2 in which cells have been positioned in locations $A-D$. If we apply the rule set to all cells, we see that only the cell in location D can move since Rules 3 and 4 must have no cells on either side of the current cell, Rule 5 must have an empty cell to the south, and Rule 1 requires an empty cell to the north and west. The result is that the cell in location D will move by using Rule 2 to location E . By the above argument, the cell originally in location D will be able to move to location L unless some other *moving* cell can interfere with its motion (for the rest of the proof, referring to a cell by a letter name means that that cell was originally at that location). Since D 's motion on the surface of the array cannot cause any cell on the surface to move, the only other moving calls must result from D 's original northeast move using Rule 2.

Indeed, D 's original move allows the cell at location C to move northward using Rule 1. At this point C is free to follow D around the surface of the cell array. If it can be shown that C cannot interfere with D 's motion, then D will arrive at location L . For C to interfere with D 's motion, C would have to prevent Rule 3, Rule 4, or Rule 5 from executing. C can only prevent D from executing Rule 3 or Rule 4 if C can move to the location immediately west of D . But C can only move eastward via Rules 2 or 3 which guarantee 2 free locations to the east, so neither Rule 2 or Rule 3 can move C to a location one location west of D . And, since Rule 5 has no preconditions to the north or east, C would have to move in front of D to interfere with D 's motion. However, one cell can only move in front of another using Rule 2 followed by Rule 4, and this could only occur on the north edge of the cell array. But we have shown that no cell can move into a location which would allow Rule 2 to be satisfied on the north face, so no moving cell on the surface of a rectangular cell array can ever "jump over" another moving cell on the surface of a rectangular cell array.

Thus we have shown that the cell originally at location D must move to location L . By using the same argument it can be shown that the cell originally at location C must move to location K unless D can interfere with C 's movement. While D is moving east along the north edge of the cell array it can prevent C from moving by not moving itself, but D and C cannot interact as shown above. While C and D are moving south along the east edge D can again prevent C from moving by not moving itself, but C must stay to the north of D since Rule 4 cannot be applied due to the presence of a cell on C 's west face. Similarly, cells moving north on the west edge of the cell array must maintain their single file order. Therefore, the only possibilities for multiple moving cells to interact are at the corners of the cell array. At the northwest corner there is no problem because a cell can only execute Rule 2 when there is enough free space between it and some other cell on the north edge of the cell array so that after Rule 2 has been applied there is still free space between the two cells. At the northeast corner, however, there is a possibility for moving cells to interact. It will be shown in Lemma 3 that this interaction could theoretically lead to the disconnection of the trailing cell, but that the random round-robin simulation model we are using prevents disconnection from happening. Here we will assume the disconnection does not happen and therefore, cell C will reach location K .

The cells at locations B and A will follow in order, assuming that the cell at location M cannot move. M must be the next cell to move other than $A-D$ for the same reason D was the only cell that could move originally. The issue of whether M could move before A has passed by M 's north face is dependent on the amount of delay cells might have in occupying locations which the rules allow. For the moment, we are assuming the random round-robin simulation model described above which puts a bound on the relative intercell actuation delay. Given this assumption, M cannot move before A has passed by M 's north face because before D can move to location H , the prerequisite for M to move, C has moved to location D , preventing M 's move. Likewise for cells B and A . The result is that M cannot move until the cell originally at A is at location H . At that point M will move, starting a new cycle of motion which will create a new column on the eastern edge of the cell array. Thus we have shown that the movement is cyclic and will continue without bound and, therefore, a rule will always apply as long as the structure remains connected. \square

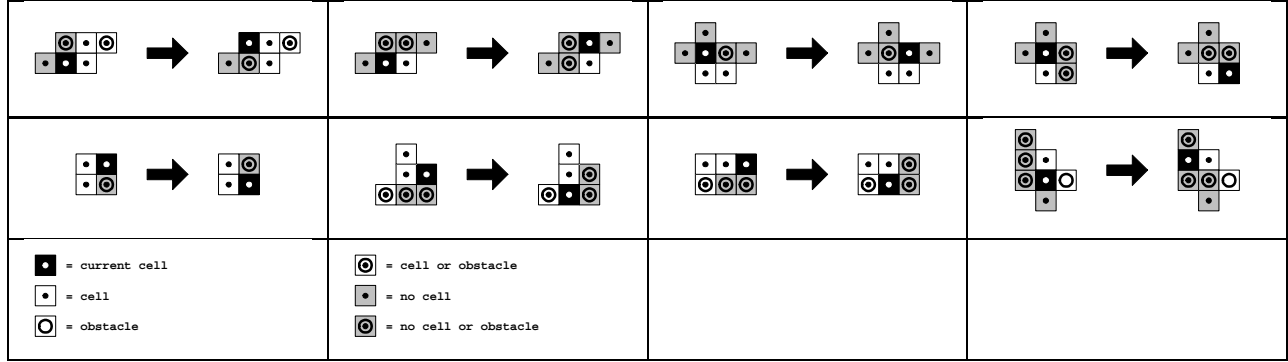


Figure 3: Eight rules for eastward locomotion with obstacles.

Lemma 2 *Given a rectangular cell array, eastward motion must result from all possible sequences of continuous rule activations from the rule set shown in Fig. 1.*

Proof: By examining the rules in Fig. 1 it can be seen that Rule 1 is a northward moving rule, Rule 5 is a southward moving rule, and Rules 2-4 have an eastward component of motion. There is no rule which produces westward motion. Since by Lemma 1 some rule can always be applied, the cell array will move eastward as long as there are no oscillations between complementary rules. Only Rules 1 and 5 are complementary, i.e. have offsetting motions. But Rule 1 can only activate on the western edge of the cell array and Rule 5 can only activate on the eastern edge of the cell array. Therefore no oscillation can occur, and the cell array must move eastward. \square

Lemma 3 *Given a rectangular cell array, no possible sequence of rule activations from the rule set shown in Fig. 1 can disconnect the cell array.*

Proof: By examination of the rule set it can be seen that no rule applied to a single cell on the surface of a cell array can cause that cell to become disconnected from the cell array, because every rule contains as a prerequisite the existence of a post-motion connecting cell. Thus, a disconnection can only occur as a result of some configuration of multiple moving cells. As shown in Lemma 1, the only possible location for interaction which results in more than simple delay is at the northeast corner of the cell array. Here, a cell moving from location H to location I (see center diagram of Fig. 2) satisfies the preconditions of Rule 3 for a trailing cell, allowing the trailing cell to move from location H to N . At this point, if the cell at location I moves to location J via Rule 5, the cell at location N will be disconnected (see right diagram of Fig. 2). The other possibility is that the cell at N moves to location O before the cell at location I moves, but the result is the same, a disconnection after the cell at location I moves to location J (note that no rules apply to a cell at location O).

Although a disconnection is theoretically possible using this rule set, it would require the trailing cell to move at least three times between the move of the leading cell from location H to location I and the move of the leading cell from location I to location J . This is because, at the time when the leading cell moves from H to I , the trailing cell can be no closer to the leading cell than location F since Rule 3 enforces a single empty cell location between moving cells on the north edge of the cell array. Thus, after the leading cell has moved from H to I , the trailing cell would have to apply Rule 3 three times to be at location N (four moves would be required to reach location O) before the move of the leading cell which disconnects the trailing cell. However, the random round-robin simulation model we are using does not allow one cell to move more than twice between the moves of another cell. Therefore, given our simulation model the cell array must remain connected. \square

Theorem 4 *The rule set in Fig. 1 produces eastward locomotion on a rectangular array of cells.*

Proof: Lemmas 1,2, and 3 prove that some rule from the rule set can always be applied to the cell array, eastward motion must result from all possible sequences of rule activations, and the array remains connected. Therefore, the rule set produces eastward locomotion on a rectangular array of cells. \square

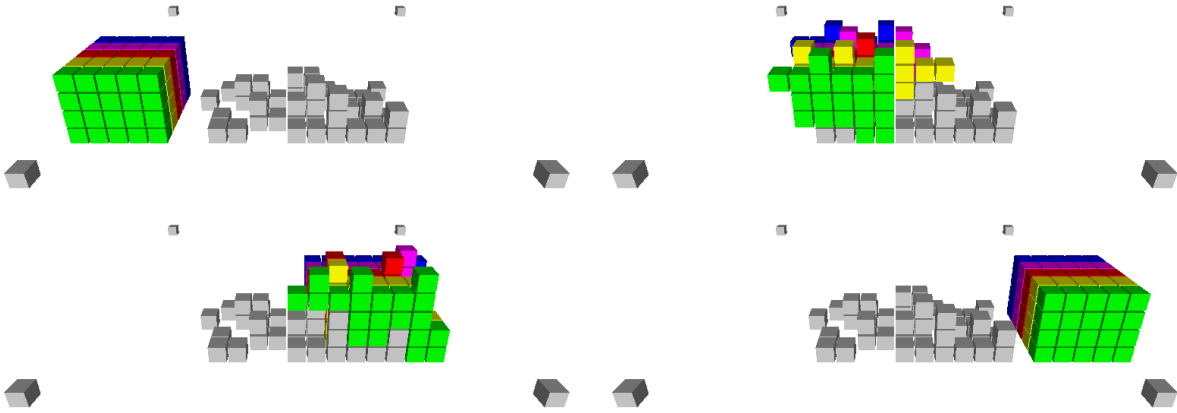


Figure 4: Four snapshots of a simulation using the rule set in Fig. 3. Each of the 5 layers in the z axis are independently running the rule set.

Locomotion with obstacles It is possible to augment the rule set in Fig. 1 to allow the cell array to crawl over obstacles. The new rule set, Fig. 3, has eight rules. The three new rules allow for southwest and northwest cell movement which is needed for the cells to comply to the obstacle field (two rules are used to implement the southwest movement). Fig. 4 shows four snapshots from a simulation run of the rules set in Fig. 3. In this case, the robot is composed of five separate layers, each running the rule set independently. The observed motion of the robot is very compliant to the terrain and, at high speed, appears to flow over the obstacles. For this set of rules, the maximum obstacle height is one less than the height of the cell array.

A proof that this rule set produces correct locomotion over any obstacle field follows much the same reasoning as the previous proof. This analysis assumes that the group of cells begins in an area free of obstacles. Regardless of the exact structure of the obstacles, the cells of the automata will still always have a connected northern row (with some moving cells above this row). This ensures that cells can continue to move from the back to the front of the group, with one rule applying at all times. To show forward progress, we show that the addition of the NW and SW motions does not induce loops in the actuation of a single cell, as each motion can be executed only once while the cell moves from the back of the group to the front. Finally, showing that disconnection does not occur is the most challenging aspect of the proof, since the mass of cells may be only one cell thick over a large obstacle. However, we can show that when the western edge of the group is isolated by an obstacle to its east, the cell at the SW corner will move first, and the cells will leave from this corner one at a time without causing disconnections. The NE and NW corners of the group will also maintain connection even when only one cell thick, since the necessary presence of moving cells (due to the random round-robin model) in their vicinity will prevent any rules from activating for the corner cells.

Second set of rules In order to climb over a higher obstacle, another set of rules can be used. It is composed of 22 rules. It uses only the Moore neighborhood but uses two states of units: activated and inactivated. Only activated units are permitted to move. The rules are classified into three types: five activation rules, 12 moving rules for activated units, and five inactivation rules. Two examples of simulated sequences using these rules are shown in Fig. 5. Gray, dark gray, and black squares represent an obstacle, an inactivated unit, and an activated unit, respectively. Correctness of the rule set has not yet been determined, but simulation studies give good results. In addition, we are investigating techniques to automatically switch from one rule set to another based on the local form of the environment.

Instantiations Once these algorithms have been developed in the abstract, it is our intent that they be realized on a variety of self-reconfiguring robot systems. This primarily requires developing actuation primitives for the system that can realize the types of motion called for by the cellular automata. One system that we are currently investigating is that of Murata *et al.*[MYT⁺00]. This system is quite powerful, but the modules are somewhat unusual, as can be seen in Fig. 6. First of all, a module cannot connect to neighbors on all sides, as is assumed in

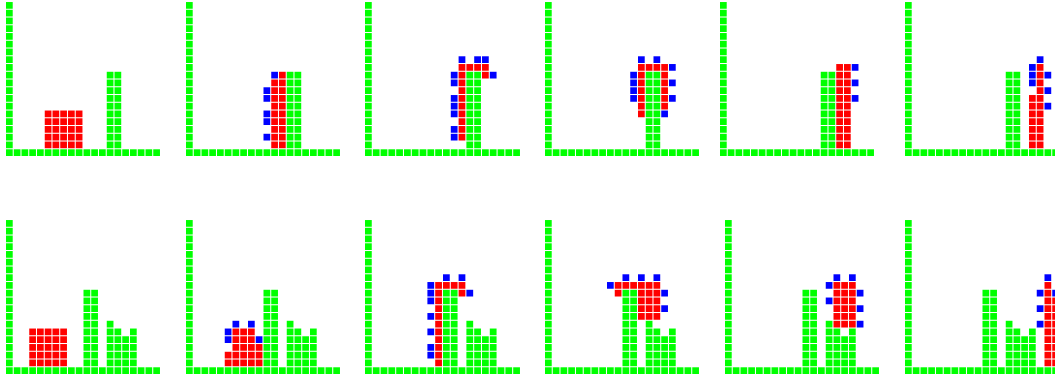


Figure 5: Snapshots of two different simulations based on the 21 rule set.

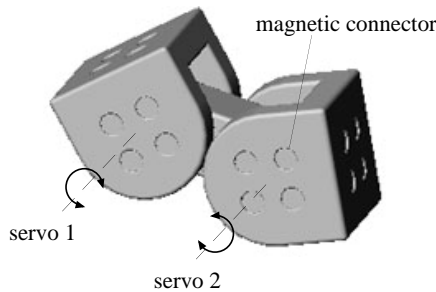


Figure 6: One module of the system of [MYT+00].

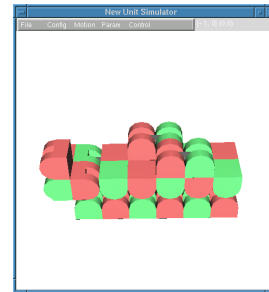


Figure 7: Screenshot of the simulated instantiation of the five rule set.

the cellular automata. This requires that at least two 2-D layers be present to maintain connections between all modules. In addition, each module consists of two articulated bodies, and so inhabits a configuration space larger than the two dimensions of the simple automata described here. However, we have had initial success in applying the five rule set to this system. Neighbor relationships translate in a straightforward way, and movement of a module along a straight line is also easy. For turning around corners, we have had to add the concept of a helper, and explicit message passing, since a module will need its neighbor to perform the actuation to make the northeast and southeast transitions. Proper message passing is also required to ensure that the group remains connected despite the limited connection ability of the individual modules. A distributed simulation has been implemented as the instantiation of these rules and has produced successful forward locomotion. The results of this simulator can be displayed in the GUI developed for these modules by Murata’s group, as shown in the screenshot in Fig. 7.

Another way in which these automata may be easily instantiated onto several different systems is through the use of meta-modules. For many systems, such as the Molecule, it is possible to construct a group of several non-isotropic modules in such a way that the group has isotropic motion capability. For the Molecule, a structure called a *tile* has been developed with this property [KR00]. These meta-modules can then immediately use the rules of the automata, although communication between the individual modules would be necessary to determine the meta-module’s neighbors and coordinate its actuation. Also, the use of meta-modules requires many more modules to achieve the same amount of reconfigurability. In the future we also hope to instantiate at least one set of rules onto the Molecule [KR99] and Crystalline Atom [RV01], each of which have their own unique actuation properties, either directly or through the use of meta-modules.

References

- [BBR] Z. Butler, S. Byrnes, and D. Rus. Distributed motion planning for modular robots with unit-compressible modules. Submitted to IROS 2001.

- [FK90] T. Fukuda and Y. Kawakuchi. Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 662–7, 1990.
- [KR99] K. Kotay and D. Rus. Locomotion versatility through self-reconfiguration. *Robotics and Autonomous Systems*, 26:217–32, 1999.
- [KR00] K. Kotay and D. Rus. Scalable parallel algorithm for configuration planning for self-reconfiguring robots. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, Boston, 2000.
- [MKY⁺98] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, and S. Kokaji. A 3-D self-reconfigurable structure. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 432–9, May 1998.
- [MYK⁺01] S. Murata, E. Yoshida, H. Kurokawa, K. Tomita, and S. Kokaji. Self-repairing mechanical systems. *Autonomous Robots*, 10:7–21, 2001.
- [MYT⁺00] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, and S. Kokaji. Hardware design of modular robotic system. In *Proc. of the Int'l Conf. on Intelligent Robots and Systems*, pages 2210–7, 2000.
- [PCSC96] A. Pamecha, C-J. Chiang, D. Stein, and G. Chirikjian. Design and implementation of metamorphic robots. In *Proc. of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, 1996.
- [RV01] D. Rus and M. Vona. Crystalline robots: Self-reconfiguration with unit-compressible modules. *Autonomous Robots*, 10(1):107–24, 2001.
- [SWC99] W.-M. Shen, P. Will, and A. Castano. Robot modularity for self-reconfiguration. In *SPIE Conf. on Sensor Fusion and Decentralized Control in Robotic Systems 2*, 1999.
- [TMK⁺99] K. Tomita, S. Murata, H. Kurokawa, E. Yoshida, and S. Kokaji. Self-assembly and self-repair method for a distributed mechanical system. *IEEE Trans. on Robotics and Automation*, 15(6):1035–45, Dec. 1999.
- [UK00] Cem Ünsal and Pradeep Khosla. Mechatronic design of a modular self-reconfiguring robotic system. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 1742–7, 2000.
- [YDR00] M. Yim, D. Duff, and K. Roufas. PolyBot: a modular reconfigurable robot. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, 2000.
- [YMK⁺00] E. Yoshida, S. Murata, A. Kaminura, K. Tomita, H. Kurokawa, and S. Kokaji. Motion planning of self-reconfigurable modular robot. In *Proc. of Int'l Symposium on Experimental Robotics*, 2000.