Linear Reconfiguration of Cube-Style Modular Robots

Greg Aloupis[*] Sébastien Collette[†] Mirela Damian[‡] Erik D. Demaine[§]
Robin Flatland[¶] Stefan Langerman[‖] Joseph O'Rourke[**]
Suneeta Ramaswami[††] Vera Sacristán[‡‡] Stefanie Wuhrer[§§§§]

Abstract

In this paper we propose a novel algorithm for both contracting and expanding cube-style modular robots which reconfigures any given source robot composed of n atoms into any given target robot with $O(n)$ basic actions, improving on previous $O(n^2)$ algorithms. During reconfiguration, the robot forms one connected component at all times, and the reconfiguration takes place within the union of the bounding boxes of the initial and final robot configurations. The algorithm can be implemented using distributed control and allowing massive simultaneous parallel moves of the atoms.

1 Introduction

Self-reconfiguring robotic systems have received increased interest. When operating under uncertainty of environmental models and/or task specifications, self-reconfiguring modular robots have the ability to adapt to restricted environments such as narrow tunnels or rough terrains, circumvent or climb over obstacles such as walls or stairs, grow emergency structures such as bridges, surround and manipulate objects in outer space, inspect mechanisms and unaccessible spaces e.g., in nuclear plants, etc. Self-reconfiguring modular robots are self sufficient systems that cannot only change their shape, but are also capable of locomotion and self-repairing.

Modular robots are composed of simple units, very limited in their actions, that can be assembled together to form versatile robotic systems. When all the units are identical, the modular robot is called homogeneous. Units for modular robots have been studied and prototyped that can perform a wide variety of basic actions. In this paper we consider self-reconfiguring modular robots composed of cubical units (atoms) arranged in a lattice configuration and capable of performing four basic actions: expand and contract, as well as attach and detach from neighbors. The atoms are arranged in meta-modules, groups of atoms attached to one another in a cubic shape, that are known to be necessary and sufficient to reconfigure any robot.

When reconfiguring, several properties are desirable for these systems. One is that the units that form the robot are autonomous, and the reconfiguration can be performed under distributed control,

[*]Université Libre de Bruxelles, Belgique, greg@vision.scs.carleton.ca. Supported by the Communauté Française de Belgique - Actions de Recherche Concertées (ARC).

[†]Aspirant du FNRS, Université Libre de Bruxelles, Belgique, sebastien.collette@ulb.ac.be. Supported by the Communauté Française de Belgique - Actions de Recherche Concertées (ARC).

[‡]Villanova University, Villanova, USA, mirela.damian@villanova.edu.

[§]Massachusetts Institute of Technology, Cambridge, USA, edemaine@mit.edu. Partially supported by NSF CAREER award CCF-0347776 and DOE grant DE-FG02-04ER25647.

[¶]Siena College, Loudonville, N.Y., USA, flatland@siena.edu

[‖]Chercheur Qualifié du FNRS, Université Libre de Bruxelles, Belgique, stefan.langerman@ulb.ac.be. Supported by the Communauté Française de Belgique - Actions de Recherche Concertées (ARC).

[**]Smith College, Northampton, USA, orourke@cs.smith.edu.

[††]Rutgers University, Camden, USA, rsuneeta@camden.rutgers.edu. Partially supported by NSF grant CCR-0204293.

[‡‡]Universitat Politècnica de Catalunya, Barcelona, Spain, vera.sacristan@upc.edu. Partially supported by projects MEC MTM2006-01267 and DURSI 2005SGR00692.

[§§§§]Carleton University, Ottawa, Canada, swuhrer@scs.carleton.ca.

independent from any central processor. Parallel reconfiguration algorithms are desirable, because simultaneous unit movements make the reconfiguration faster. Finally, when space restrictions apply, in-place reconfiguration is required or, at least desirable, so that the robot does not expand outside its initial and final shapes or, at least, does not expand outside their bounding boxes. All the reconfiguring moves must guarantee that the robot stays connected at all times. Finally, it is desirable that the reconfiguration space is connected.

Our main contribution is a novel algorithm that satisfies all these requirements and reconfigures any given source robot composed of n atoms and arranged in meta-modules into a given target robot with $O(n)$ basic atom actions, within the union of the bounding boxes of the initial and final robot configurations. Previously, only $O(n^2)$ algorithms were known [RV01, VYS02, BR03]. Throughout the paper, n refers to the number of robot atoms.

2 Definitions and basic moves

An *atom* is a cubical device equipped with an expansion/contraction mechanism that allows it to extend its faces out and retract them back. When two opposing faces are extended, the length of the atom along that dimension is twice what it is when the two faces are retracted. Each face is equipped with an attaching/detaching mechanism that allows atoms to lock face-to-face with adjacent atoms. We list below the four primitive operations an atom is capable of performing, where direction is one of x^+, x^-, y^+, y^-, z^+ or z^-:

CONTRACT(***direction***)**.** Compress the atom in the indicated direction.

EXPAND(***direction***)**.** Expand the atom in the indicated direction.

ATTACH(***direction***)**.** Attach the atom to its neighbor in the indicated direction.

DETACH(***direction***)**.** Detach the atom from its neighbor in the indicated direction.

A *robot* is a connected collection of atoms arranged in a three dimensional lattice. The collection is connected in the sense that its dual graph (vertices correspond to atoms, edges correspond to attached atoms) is connected. A robot can reconfigure itself by having groups of its atoms perform the primitive operations in a coordinated way, resulting in atoms moving relative to one another.

Two existing hardware models implement these robots: crystalline robots, which are contracting cube-style modular robots, and telecube robots, which are expanding cube-style modular robots.

Contracting cube-style modular robots: Crystalline atoms [RV01] have an expanded natural position, and their movement is based on the fact that they can contract.

Expanding cube-style modular robots: Telecube atoms [VYS02] have a contracted natural position, and their movement is based on the fact that they can expand.

There exist atom configurations which cannot be reconfigured. For example, in a linear arrangement of atoms, no atom can change its relative position with respect to the remaining atoms without disconnecting the robot. Connected lattice arrangements of what are known as meta-modules, however, guarantee connectedness of the reconfiguration space [RV01, VYS02]. A *meta-module* is a group of atoms attached to one another in a cubic shape. The number of atoms necessary for a meta-module depends on the hardware model considered. Crystalline meta-modules are composed of $4 \times 4 \times 4$ atoms, which are shown necessary and sufficient to reconfigure any robot [RV01]. Telecube meta-modules are composed of $2 \times 2 \times 2$ atoms, which are shown necessary and sufficient to reconfigure any robot [VYS02]. We define two basic moves (hardware independent) that meta-modules are able to perform. The reconfiguration algorithms to be described in Section 3 use these moves extensively.

SLIDE(***dirSlide***)**.** Slide the meta-module one step in the direction *dirSlide* with respect to some substrate meta-modules. This move is illustrated in Figure 1. The preconditions for applying this move are:

(i) The sliding meta-module (A in Fig. 1a) is adjacent to a meta-module in a direction orthogonal to *dirSlide* (B in Fig. 1a), which in turn is adjacent to a meta-module in direction *dirSlide* (C in Fig. 1a).

(ii) The target position for the sliding meta-module is free.

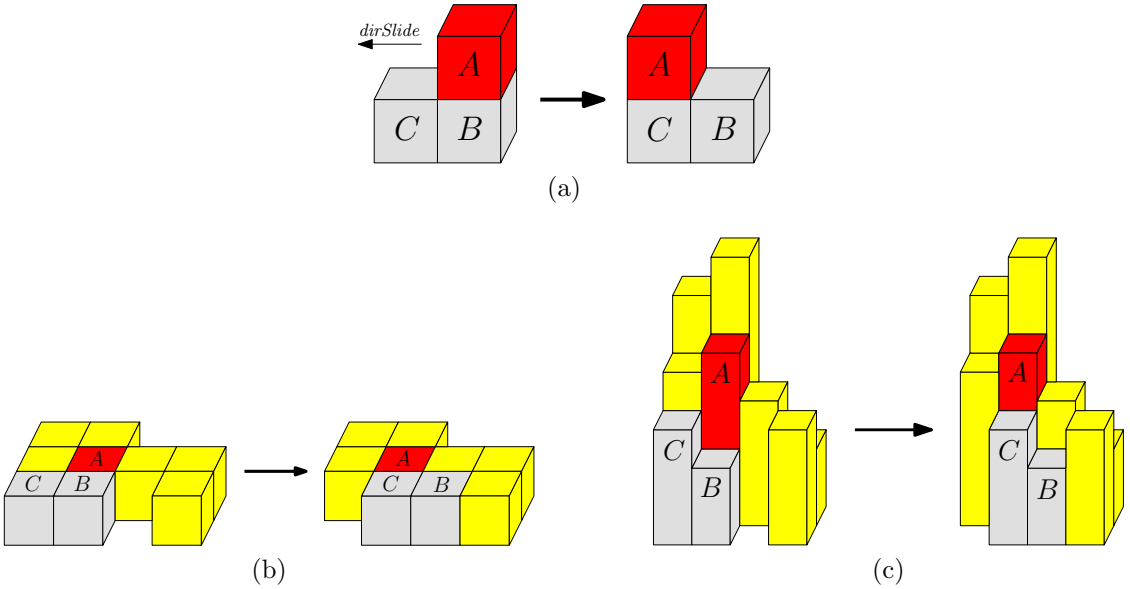

Figure 1: Examples of SLIDE(x^-): (a) Meta-module A slides alone (2D projection), (b) A carries adjacent meta-modules (2D projection), (c) A carries towers.

This move allows the sliding meta-module to "carry" other attached meta-modules (as in Figure 1b), as long as the target position for a carry-on meta-module is unoccupied. Similarly, this move allows an entire tower of meta-modules sitting on top of the sliding meta-module to be carried along, as shown in Figure 1c.

k-TUNNEL(***sPos*, *ePos***). Push the meta-module located at *sPos* (start position) into the robot, and pop a meta-module out of the robot in position *ePos* (end position). The preconditions for applying this move are:

(i) *sPos* is at a leaf node in the dual graph of the starting configuration, and *ePos* is a leaf node in the dual graph of the ending configuration.

(ii) There is an orthogonal path through the robot starting at *sPos* and ending at *ePos*, with k orthogonal turns.

This move is illustrated in Figure 2 for $k = 1, 2, 3, 4$. Although the k-TUNNEL move is defined for arbitrary k, our reconfiguration algorithms only require $k \leq 4$.

In both contracting and expanding cube-style modular robots, the SLIDE move can be achieved by repeating a constant number of times a finite sequence of primitive atom operations known as the INCHWORM action [RV01, VYS02]. Details appear in the full version.

Lemma 2.1. SLIDE *can be implemented using $O(1)$ primitive actions in both the expanding and the contracting models, without disconnecting the robot.*

Similarly, any k-TUNNEL move can be achieved by alternate application of $k+1$ TRANSFER actions and k TURN actions. Both TURN and TRANSFER consist of a finite sequence of primitive atom operations in both the contracting and the expanding models. Such sequences have been used in previous reconfiguration algorithms [RV01, VYS02]. Details appear in the full version.

Lemma 2.2. k-TUNNEL *can be implemented using $O(k)$ primitive actions in both the expanding and the contracting models, without disconnecting the robot.*

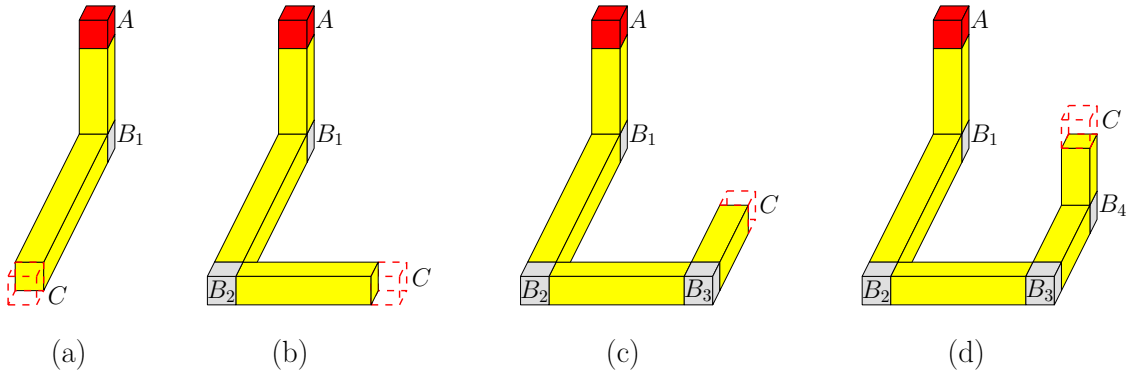

Figure 2: Examples of TUNNEL(A, C) with orthogonal turns at B_i, $i = 1, 2, 3, 4$ (a) 1-TUNNEL (b) 2-TUNNEL (c) 3-TUNNEL (d) 4-TUNNEL.

3 Centralized Reconfigurations

In this section we present a centralized algorithm that reconfigures any given source robot, S, into any given target robot, T, in $O(n)$ basic moves that are performed "in place". By "in place" we mean within the union of the bounding boxes of the source and target robots, assuming that they overlap.

In Section 3.1 we consider the simpler case of robots consisting of single layers of meta-modules, which we will refer to as *2D robots*. We will then discuss the general case of arbitrary robots in Section 3.2.

3.1 Centralized Reconfiguration in 2D

The main idea behind the algorithm is to specify a simpler shape, which we refer to as a "2D comb," that an arbitrary robot can reconfigure to and from. We then show that any 2D comb can be transformed into any other 2D comb.

3.1.1 2D Robot to 2D Comb

Here we describe an algorithm for reconfiguring S, an arbitrary 2D configuration of robot meta-modules, into an intermediate *comb* configuration [ABMP06], C_S. In a comb configuration, the meta-modules form a special type of histogram polygon [CSW99]. Specifically, the meta-modules are arranged in horizontally adjacent columns, with the bottom meta-module of each column in a common row which is called the *handle*; the columns of meta-modules extending upward from the handle are called *teeth*.

Initially, the algorithm designates the row containing the topmost meta-modules of S as the *wall*. We view the wall as infinite in length. The wall sweeps the entire robot configuration, moving downward by one row of meta-modules in each step. By having certain meta-modules SLIDE downward with the wall, the teeth of the comb emerge upward from the wall. We call this process "combing" the robot. Algorithm 3.1 describes our 2D-COMBING method.

Algorithm 3.1. 2D-COMBING(S)

1. Set wall to row containing topmost meta-modules of S.
2. **while** there are meta-modules below the wall **do**
   (a) Label wall meta-modules moving or stationary.
   A meta-module within the wall is labeled *stationary* if there is a meta-module adjacent below; otherwise, the meta-module is labeled *moving*.

(b) Identify moving wall components.
A *moving wall component* is a maximal wall component of adjacent moving wall meta-modules. Observe that a moving wall component always has a stationary meta-module adjacent to one or both ends; otherwise, the moving component would be disconnected from the rest of the object.

(c) Move wall one row lower, carrying along moving wall components and their attached teeth. Each moving wall meta-module adjacent to a stationary meta-module performs a SLIDE(y^-) move, thus moving itself one row below w.r.t. the adjacent stationary wall meta-module. In the process, the sliding meta-module carries along the entire moving component it belongs to, which also pulls along the attached teeth (if any).

(d) Adjust meta-module attachments
   i. Attach adjacent wall meta-modules.
   ii. Detach meta-modules in w^+ from meta-modules adjacent to left (x^-) and right(x^+).
   iii. Attach meta-modules in w^- to wall meta-modules newly adjacent above (y^+ direction).

Lemma 3.2. *The* 2D-COMBING *algorithm reconfigures S to C_S using $O(n)$ moves. The robot configuration forms one connected component at all times, and the reconfiguration is done in place.*

3.1.2 2D Comb to 2D Comb

In this section, we describe an algorithm to reconfigure any comb robot C_S to any other comb robot C_T. Assume that the left end meta-modules of C_S and C_T are aligned. The basic idea behind our algorithm is to lengthen short teeth in C_S by moving meta-modules from long teeth in C_S. This is done one meta-module at a time by using 1-TUNNEL or 2-TUNNEL moves until the short tooth has the required length in C_T. Furthermore, the short teeth are lengthened in left to right order by moving meta-modules from long teeth in right to left order. The details of our algorithm are given in Algorithm 3.3. We represent meta-modules by their (i, j) coordinates in the lattice.

Algorithm 3.3. 2D-COMB-TO-COMB(C_S, C_T)

1. Let $O \leftarrow \{(i,j) \mid (i,j) \in C_S$ **and** $(i,j) \notin C_T\}$. Sort O in decreasing lexicographic order.
2. Let $I \leftarrow \{(i,j) \mid (i,j) \in C_T$ **and** $(i,j) \notin C_S\}$. Sort I in increasing lexicographic order.
3. **while** $|I| > 0$ (that is, short teeth remain)
   (a) $oPos \leftarrow \min\{O\}$.
   (b) $iPos \leftarrow \min\{I\}$.
   (c) 1-TUNNEL($oPos$,$iPos$) (if $y(oPos) = 1$ or $y(iPos) = 1$), or 2-TUNNEL($oPos$,$iPos$).
   (d) Delete $oPos$ from O.
   (e) Delete $iPos$ from I.

Lemma 3.4. *The* 2D COMB-TO-COMB *algorithm reconfigures C_S to C_T using $O(n)$ moves. The robot configuration forms one connected component at all times, and the reconfiguration is done in place.*

3.1.3 Overall 2D Reconfiguration Algorithm

The general algorithm to reconfigure any robot S to any other robot T consists of three major steps:

1. Apply the 2D-COMBING algorithm to reconfigure source S into source comb configuration C_S.
2. Apply the 2D COMB-TO-COMB algorithm to reconfigure source comb C_S into target comb C_T.
3. Apply the moves of the 2D-COMBING algorithm in reverse order to reconfigure target comb C_T into target robot T.

Theorem 3.5. *Any source robot can be reconfigured into any target robot using $O(n)$ moves. The robot configuration forms one connected component at all times, and the reconfiguration is done in place.*

3.2 Centralized Reconfiguration in 3D

The main idea behind the 3D reconfiguration algorithm is analogous to the 2D case: specify a simpler shape, which we refer to as a "comb of combs" (3D comb for short), that an arbitrary robot can reconfigure to and from. We also show that any 3D comb can be reconfigured into any other 3D comb. The reconfiguration into the 3D comb goes through another intermediate shape, which we call a "3D terrain". In the following we describe each reconfiguration step.

3.2.1 Source Robot to 3D Terrain

We use the 3D analog of the 2D-Combing process, 3D-Combing, to reconfigure S into a 3D terrain as follows. The wall now consists of an entire 2D horizontal layer of meta-modules, initially the topmost single layer of S. In each step of the algorithm, the wall moves downward through S by one layer. At any given step, the wall consists of a collection of meta-modules, some of which are stationary and some of which are moving. Analogous to the 2D case, a stationary meta-module is one that has an adjacent meta-module below. Unlike the 2D case, where a 2D moving wall component is one row of adjacent meta-modules, a 3D moving wall component is an arbitrarily shaped maximal component of adjacent moving meta-modules within the wall. When the wall moves downward by one layer, the wall moves past the stationary meta-modules, while the moving meta-modules get carried along with the wall using Slide(z^-). The robot stays connected at all times. The final result is that all meta-modules of S having the same (x, y) coordinates are grouped together to form a contiguous tower of meta-modules, and the resulting robot consists of a set of towers extending in the z^+ direction and resting on an arbitrarily-shaped layer (in the xy-plane) of connected meta-modules.

Lemma 3.6. *The* 3D-Combing *algorithm reconfigures any robot into a 3D terrain using $O(n)$ moves. The robot configuration forms one connected component at all times, and the reconfiguration is done in place.*

3.2.2 3D Terrain to 3D Comb

In this section we reconfigure a 3D Terrain I (here I stands for Intermediate configuration) into a 3D comb by applying a 3D-Terrain-To-Comb algorithm which reconfigures the base of I into a 2D comb, without altering the towers resting on the base of I. This can be accomplished using the 2D-Combing algorithm of Section 3.1.1. Recall that the Slide move allows meta-modules to carry towers, as illustrated in Figure 1c. This implies that the 2D-Combing algorithm applies not only to single layer robots, but also to robots consisting of a single layer on with towers sitting on top.

After this second combing pass, the resulting robot is a 2D comb whose teeth are combs themselves, i.e., each tooth itself is a the handle of a comb with teeth extending in the z-direction; hence, the name of "comb of combs", or 3D comb for short.

Lemma 3.7. *The* 3D-Terrain-To-Comb *algorithm reconfigures any 3D terrain into a 3D comb using $O(n)$ moves. The robot configuration forms one connected component at all times, and the reconfiguration is done in place.*

3.2.3 3D Comb to 3D Comb

In this section we describe an algorithm 3D-Comb-To-Comb to reconfigure any source 3D comb C_S into any given target 3D comb C_T. Both C_S and C_T have the configuration described in the previous section; that is, they consist of a single comb in the xy-plane (call this the xy-comb), and each tooth of the xy-comb is itself the handle of a comb with teeth extending in the z direction (call these the z-combs). Let S_{xy} (T_{xy}) denote the xy-comb of C_S (C_T). Suppose that the handles of S_{xy} and T_{xy} are aligned with matching left end meta-modules.

Let $S_1, S_2, \ldots, S_m$ be the z-combs of C_S. Similarly, let $T_1, T_2, \ldots, T_\ell$ be the z-combs of C_T. Let $|S_i|$ ($|T_i|$) be the number of meta-modules in S_i (T_i). If $m \geq \ell$ (that is, the handle of S_{xy} is no shorter

than the handle of T_{xy}), we consider $|T_i| = 0$ for all $i > \ell$. Similarly, if $\ell > m$ (that is, the handle of T_{xy} is longer than the handle of S_{xy}), we consider $|S_i| = 0$ for all $i > m$. For each $S_i, 1 \leq i \leq \max(m, \ell)$, define $L(S_i)$ to be $|S_i| - |T_i|$. Call a z-comb S_i of C_S *large* (*small*) if $L(S_i) > 0$ ($L(S_i) < 0$); in this case, S_i has more (less) meta-modules than T_i and some meta-modules will need to be moved out of (into) S_i.

The basic idea behind our algorithm to reconfigure C_S to C_T consists of the following two basic steps:

1. First enlarge a small z-comb S_i of C_S by moving meta-modules from a large z-comb $S_{i'}$ of C_S. This is done one meta-module at a time by using k-Tunnel ($k \in \{1, 2, 3, 4\}$) moves until S_i has the required size in C_T (that is, $L(S_i)$ becomes 0). A meta-module from $S_{i'}$ may be tunneled to anywhere within S_i that is an occupied position in C_T and can be reached by a k-Tunnel move. The small z-combs are enlarged in left to right order by shrinking large z-combs in right to left order.
2. Once $L(S_i) = 0, 1 \leq i \leq \max(m, \ell)$, the number of z-combs in the reconfigured S_i is equal to ℓ. Let $S_1, S_2, \ldots, S_\ell$ refer to the new z-combs of the reconfigured S. Even though $|S_i| = |T_i|$, it is possible that the configuration of S_i is different from that of T_i. In this case, reconfigure S_i to T_i by using Algorithm 2D-Comb-To-Comb.

Lemma 3.8. *The* 3D-Comb-To-Comb *algorithm reconfigures C_S into C_T using $O(n)$ moves. The robot configuration forms one connected component at all times, and the reconfiguration is done in place.*

3.2.4 Overall 3D Reconfiguration Algorithm

The general algorithm to reconfigure an arbitrary 3D robot S to a given 3D target robot T consists of five major steps:

1. $S \rightarrow I_S$. Reconfigure S into the source 3D terrain I_S using the 3D-Combing algorithm.
2. $I_S \rightarrow C_S$. Reconfigure I_S into the source 3D comb C_S using the Terrain-To-Comb algorithm.
3. $C_S \rightarrow C_T$. Reconfigure C_S into the target 3D comb C_T using the 3D Comb-To-Comb algorithm.
4. $C_T \rightarrow I_T$. Reconfigure C_T into the target 3D terrain I_T by applying the moves of the Terrain-To-Comb algorithm in reverse order.
5. $I_T \rightarrow T$. Reconfigure I_T into the target T by applying the moves of the 3D-Combing algorithm in reverse order.

Theorem 3.9. *Any source robot can be reconfigured into any target robot using $O(n)$ moves. The robot configuration forms one connected component at all times, and the reconfiguration is done in place.*

4 Distributed Implementation

The algorithms presented in Section 3 are described assuming that there exists a controlling unit that can communicate with the atoms and control and coordinate their actions. Nevertheless, they can all be implemented in a distributed and synchronous way, while keeping the overall linear-time complexity. They require an initialization step that basically consists in sending to each atom the initial configuration, as well as its address in it, together with the final configuration. After this initial step, the robots can self-reconfigure without the help of any central controller, only based on distributed control and local communication. Furthermore, the distributed algorithms that we propose allow massive parallel actuation of atoms and modules.

Theorem 4.1. *Any source robot can be reconfigured into any target robot in $O(n)$ time in a distributed and synchronous way which allows parallel actuation of atoms and modules. The robot configuration forms one connected component at all times, and the reconfiguration is done in place.*

References

[ABMP06] Esther M. Arkin, Michael A. Bender, Joseph S.B. Mitchell, and Valentin Polishchuk. The snowblower problem. In *WAFR'06: Proc. Seventh International Workshop on the Algorithmic Foundations of Robotics*, 2006.

[BR03] Zack Butler and Daniela Rus. Distributed planning and control for modular robots with unit-compressible modules. *The International Journal of Robotics Research*, 22(9):699–715, 2003.

[CSW99] Francis Chin, Jack Snoeyink, and Cao An Wang. Finding the medial axis of a simple polygon in linear time. *Discrete Computational Geometry*, 21(3):405–420, 1999.

[RV01] Daniela Rus and Marsette Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, 2001.

[VYS02] Sergei Vassilvitskii, Mark Yim, and John Suh. A complete, local and parallel reconfiguration algorithm for cube style modular robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 117–122, 2002.