Dartmouth College
Dartmouth Digital Commons

Computer Science Technical Reports | Computer Science

10-1-1998

Abstractions for Simplifying Planning in Self-Reconfigurable Robotic Systems

Craig McGray
Dartmouth College

Daniela Rus
Dartmouth College

Follow this and additional works at: https://digitalcommons.dartmouth.edu/cs_tr

 Part of the Computer Sciences Commons

Dartmouth Digital Commons Citation

McGray, Craig and Rus, Daniela, "Abstractions for Simplifying Planning in Self-Reconfigurable Robotic Systems" (1998). Computer Science Technical Report PCS-TR98-342.
https://digitalcommons.dartmouth.edu/cs_tr/165

This Technical Report is brought to you for free and open access by the Computer Science at Dartmouth Digital Commons. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.

Abstractions for Simplifying Planning in Self-Reconfigurable Robotic Systems

Craig McGray and Daniela Rus

Technical Report PCS-TR98-342

Revised October 9, 1998

Abstractions for Simplifying Planning in Self-Reconfigurable Robotic Systems

Craig McGray Daniela Rus

Department of Computer Science
Dartmouth College
Hanover, NH 03755

phone: (603) 646 1691
fax: (603) 646 1672
{craig,rus}@cs.dartmouth.edu

Abstract

In [KRVM], we described a three-dimensional self-reconfiguring robot module called the Molecule Robot.

In this paper, we provide a system of abstractions for modules in self-reconfigurable robotic systems, and show how this system can be used to simplify the motion planning of the Molecule Robot system.

1 Introduction

A self-reconfiguring robotic system consists of a set of modules that can autonomously change their relative positions in order to change the shape of the overall system. When the modules in a self-reconfigurable robotic system are homogeneous, the system lends itself to inexpensive mass production, to scalability, and to greatly increased robustness. The ability to change shape "on-the-fly" can give a robot excellent maneuverability in constrained spaces, multiple modalities of locomotion for changing terrain, and extremely flexible manipulation strategies.

To change the shape of a self-reconfigurable robot, the component modules must execute a coordinated sequence of individual moves. Because of the presumably large number of components, the motion planning required to reconfigure from one shape to another can be complex. In [ChPa], Chirikjian, et. al. present a polynomial-time solution to the planning problem for a restricted set of self-reconfigurable systems.

There are two specific qualities of these systems that simplify the planning process. These are:

1. **Position Uniqueness: The set of possible module positions can be described by the center points of close-packed polyhedra, where each position must be either full or empty. There can be no partial overlap between positions.**

2. **Surface Traversal: If a module can move to some position on a continuous surface of the robot, then it can move to any position on that continuous surface. No other modules need change position in order to facilitate these moves.**

[MKuKo] and [PaChSCh] describe implementations of two-dimensional systems that possess both of the above two qualities.

While the planning algorithms of [ChPa] generalize to the three-dimensional case, mechanical constraints make it more difficult to produce a module with the above qualities that is capable of moving in three-dimensions.

In [KRVM], we describe a three-dimensional self-reconfiguring system whose basic module is shown in Figure 1. This system exhibits neither the position uniqueness nor surface traversal property defined above. So, the general planning problem for this system remains to be solved.

In this paper we show how to construct a three-dimensional meta-module that satisfies the position uniqueness and surface traversal properties. We do so by composing 16 Molecule Robots. The resulting meta-modules connect to one another in such a way that their centers can be described as close-packed boxes. We implement transitions that allow any one of these meta-modules to traverse any continuous surface of a configuration of other similar modules. This allows us to calculate motion plans for these modules using the algorithms of [ChPa].

In Section 2, we define the terms used in the remainder of the paper. In Section 3, we abstract the Molecule Robot into a class called K_0, that incorporates the module's geometric shape and motion capabilities. In Section 5, we show how the K_0 class can be used to implement another class of self-reconfiguring module, K_2, that satisfies both the position uniqueness and surface traversal properties. In Section 6, we summarize our results and conclude.

2 Definitions

Definition 2.1 (Module Position)
A module position is a placement and orientation in the space of discourse.

Definition 2.2 (Intersection)
Two positions are said to intersect if modules placed in those positions would have at least one point in common.[1]

Definition 2.3 (Adjacent)
Two positions are said to be adjacent if modules placed in those positions could attach.

Definition 2.4 (Lattice)
A lattice, L, is a triple $\{P, I, A\}$. P is a set of positions. I is a set of intersections over P, and A is a set of adjacencies over P.

Definition 2.5 (Transition)
A transition, t, is a source position, s, a destination position, d, and a set of preconditions that must be true for a module to change its position from s to d. These preconditions may take two forms. Either a position is required to be occupied by a module, or a position is required to be unoccupied. So, a transition is the four-tuple $t = \{s, d, E, F\}$, where E is the set of positions required to be empty, and F is the set of positions required to be full.

Definition 2.6 (Configuration)
A configuration, $C = \{L, Q\}$, is a lattice, $L = \{P, I, A\}$, coupled with a set of non-intersecting module positions, $Q \subseteq P$.

Two configurations, $C_1 = \{L, Q_1\}$ and $C_2 = \{L, Q_2\}$ are said to be *equivalent* iff there is a bijection, $B : Q_1 \leftrightarrow Q_2$, such that $\forall q_{1a}, q_{1b} \in Q_1, B(q_{1a})$ is adjacent to $B(q_{1b})$ iff q_{1a} is adjacent to q_{1b}.

A configuration is *connected* if there is a path of adjacencies between any two positions in the configuration.

We require that configurations remain connected at all times, and disallow the movement of any module whose removal would disconnect the configuration. This is called the *connectivity requirement.*

Definition 2.7 (Module Class)
A module class, $K = \{L, T\}$ is defined as a lattice, L, and a set of transitions, T, over that lattice.

Definition 2.8 (Reconfiguration)
Let $K = \{L, T\}$ be a class of module.
Let $C_1 = \{L, Q_1\}$ and $C_2 = \{L, Q_2\}$ be configurations.
K can reconfigure from C_1 to C_2 if at least one of the following conditions holds:

[1] If the space of discourse were continuous, we could define two positions to intersect if they share a finite neighborhood about at least one point.

- *C_1 is equivalent to C_2.*
- *There is a transition, $t \in T$ and a position, $q \in Q_1$ such that applying t to q yields a configuration, C_3, and K can reconfigure from C_3 to C_2.*

3 The Molecule Robot and the K_0 Class

Mechanical details of the Molecule Robot are described in [KRVM]. In this paper we discuss the Molecule Robot with regard only to a manhattan approximation of its physical shape, and to the set of transitions that it can perform in a connected configuration of other Molecule Robots. The method by which these transitions are achieved is described in [KRVM].

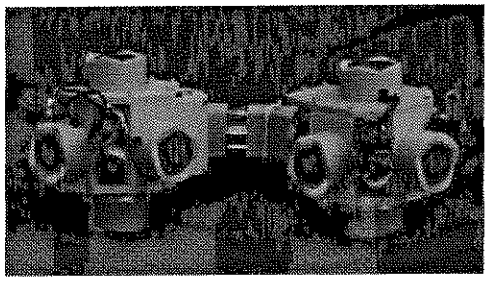

Figure 1: The Molecule Robot is an L-shaped module that can attach to other Molecule Robots at its endpoints. The endpoints (called "atoms") each have inter-module connectors on five of their six faces. The sixth face holds an intra-module connector, called a "bond". Four actuators in the Molecule Robot allow it to perform x-axis, y-axis, and z-axis rotations relative to the endpoints. These rotations may be performed only when attached to other Molecule Robots.
The module received its name from a resemblance in appearance to two atoms attached by a molecular bond.

Recall from Definition 2.7 that a *module class* is a lattice paired with a set of transitions on that lattice. We would like to describe a specific module class that can be implemented by one Molecule Robot. Let this class be $K_0 = \{L_0, T_0\}$. Note that there may be many other architectures other than the Molecule Robot that could implement the K_0 class. Conversely, there are many classes that could be implemented by the Molecule Robot.

To describe the lattice, L_0, we consider the discrete coordinate space $\mathcal{Z}^3$. Each module occupies exactly three points in an "L" shape in $\mathcal{Z}^3$: one for each "atom", and one between them for the connecting "bond".

There are 12 distinct orientations of three points arranged in this fashion. So, a position in the K_0 lattice may be described as one point in $\mathcal{Z}^3$ (its placement), along with one of 12 possible orientations. We name the orientations 1...12. So:

$$P = \{(x, y, z, r) : x, y, z \in \mathcal{Z} \text{ and } r \in \{R_1 ... R_{12}\}\}$$

We can use the distance between two points as an offset for a third point. Similarly, we can use a difference in orientations between two positions as an offset for other orientations. For example, we can say that a position is the sum:

$$p = (x, y, z, r) + ((0, 0, 0, R_5) - (0, 0, 0, R_1))$$

Note that this means "find the sequence of rotations needed to bring a body in orientation R_1 to orientation R_5," and apply that sequence to a body at (x, y, z, r). It is **not** equivalent to:

$$p = (x, y, z, r) + (0, 0, 0, R_4)$$

This notation is useful in describing the sets of adjacencies and intersections in the K_0 lattice.

Recall from Definition 2.2 that two positions intersect if two modules placed in those positions would have at least one point in common. The position $(0, 0, 0, R_1)$ intersects with the set of positions shown in Figure 2. We call this set $I_{(0,0,0,R_1)}$. There are 90 positions in $I_{(0,0,0,R_1)}$.

We can now define the set of intersections in the K_0 lattice:

$$\begin{aligned} I_0 = \{&((x, y, z, r), (x + k_x, y + k_y, z + k_z, r + k_r - R_1)) \\ &: x, y, z \in \mathcal{Z}, \\ &\quad r \in \{R_1 ... R_{12}\}, \\ &\quad \textit{and } (k_x, k_y, k_z, k_r) \in I_{(0,0,0,1)}\} \end{aligned}$$

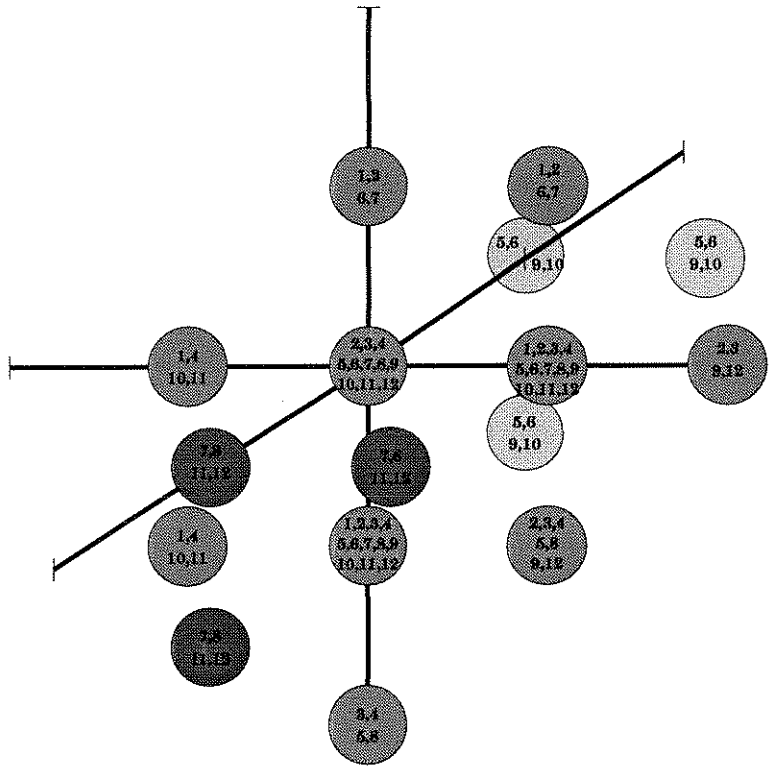

Figure 2: $I_{(0,0,0,R_1)}$: The set of intersections for position $(0, 0, 0, R_1)$. A position in the K_0 lattice is a point in $\mathcal{Z}^3$ and one of twelve possible orientations. Each node in the graph represents one point in $\mathcal{Z}^3$. The numbers in each node represent the orientations at that point that intersect with orientaion R_1 at point $(0, 0, 0)$.
Darker nodes are closer on the z-axis.

Recall from Definition 2.3 that two positions are adjacent if two modules placed in those positions could attach to one another. Any module of class K_0 can attach to a second (and is hence adjacent to the second in the K_0 lattice) if at least one of the positions at the endpoints of the module is adjacent in $\mathcal{Z}^3$ to at least one of the endpoints of the second module. The position $(0, 0, 0, R_1)$ is adjacent to the set of positions shown in Figure 3. Only non-intersecting adjacencies are shown. We call this set $A_{(0,0,0,R_1)}$. There are 219 positions in $A_{(0,0,0,R_1)}$.

We can now define the set of adjacencies in the K_0 lattice:

$$\begin{aligned} A = \{&((x, y, z, r), (x + k_x, y + k_y, z + k_z, r + k_r - R_1)) \\ &: x, y, z \in \mathcal{Z}, \\ &\quad r \in \{R_1 ... R_{12}\}, \\ &\quad \textit{and } (k_x, k_y, k_z, k_r) \in A_{(0,0,0,R_1)}\} \end{aligned}$$

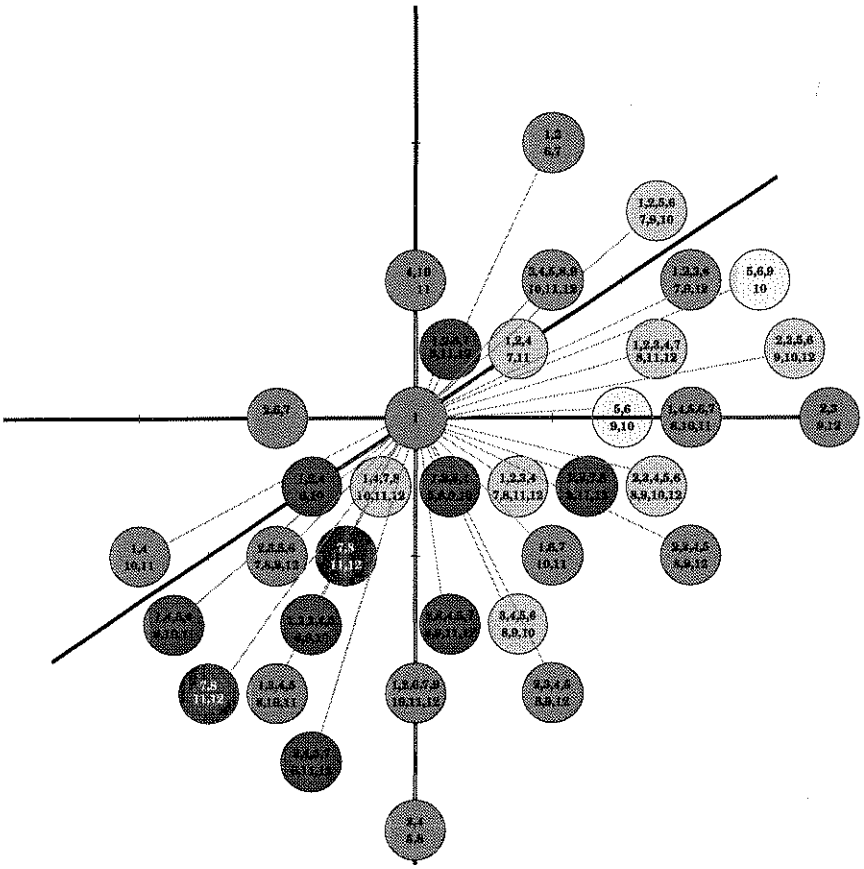

Figure 3: $A_{(0,0,0,R_1)}$: The set of adjacencies for position $(0,0,0,R_1)$. A position in the K_0 lattice is a point in $\mathcal{Z}^3$ combined with one of twelve possible orientations. Each node in the graph represents one point in $\mathcal{Z}^3$. The numbers in each node represent the orientations at that point that are adjacent to orientaion R_1 at point $(0,0,0)$. Darker nodes are closer on the z-axis.

The Molecule Robot is able to perform x, y, and z-axis rotations. These rotations are shown in Figure 4.

When a Molecule Robot performs a rotation, it can do so about either of its "atoms". Furthermore, it can rotate in either a clockwise direction or a counter-clockwise direction. So, a Molecule Robot in a given source position can rotate into one of up to 12 different destination positions. We call the set of destinations for a module at source position $(0,0,0,R_1)$, $D_{(0,0,0,R_1)}$.

In the case of the Molecule Robot, an x or z-axis rotation is possible only if some other Molecule Robot, called the *actuating module*, is attached along the axis of rotation to the "atom" about which the robot intends to rotate. (See figure 4). A y-axis rotation is possible if another Molecule Robot is attached to the atom about which the robot intends to rotate, regardless of whether or not it is lined up on the axis of rotation. The position of the actuating module is called the *actuating position*. The set of positions that could be used as the actuating position for a given rotation is called the *actuation set*.

A rotation is disallowed if it would result in a collision between two modules. So, the path travelled by a Molecule Robot during a rotation must not contain any other modules. Figure 5 shows the lattice points on the path of one of a Molecule Robot's z-axis rotations. The set of positions that are intersected during a rotation is known as that rotation's *collision set*.

For each of 12 possible destinations, $d \in D_{(0,0,0,R_1)}$, of a module at position $(0,0,0,R_1)$ we call the actuation set of d, $F_{(0,0,0,R_1)}[d]$ We call the collision set of d, $E_{(0,0,0,R_1)}[d]$.

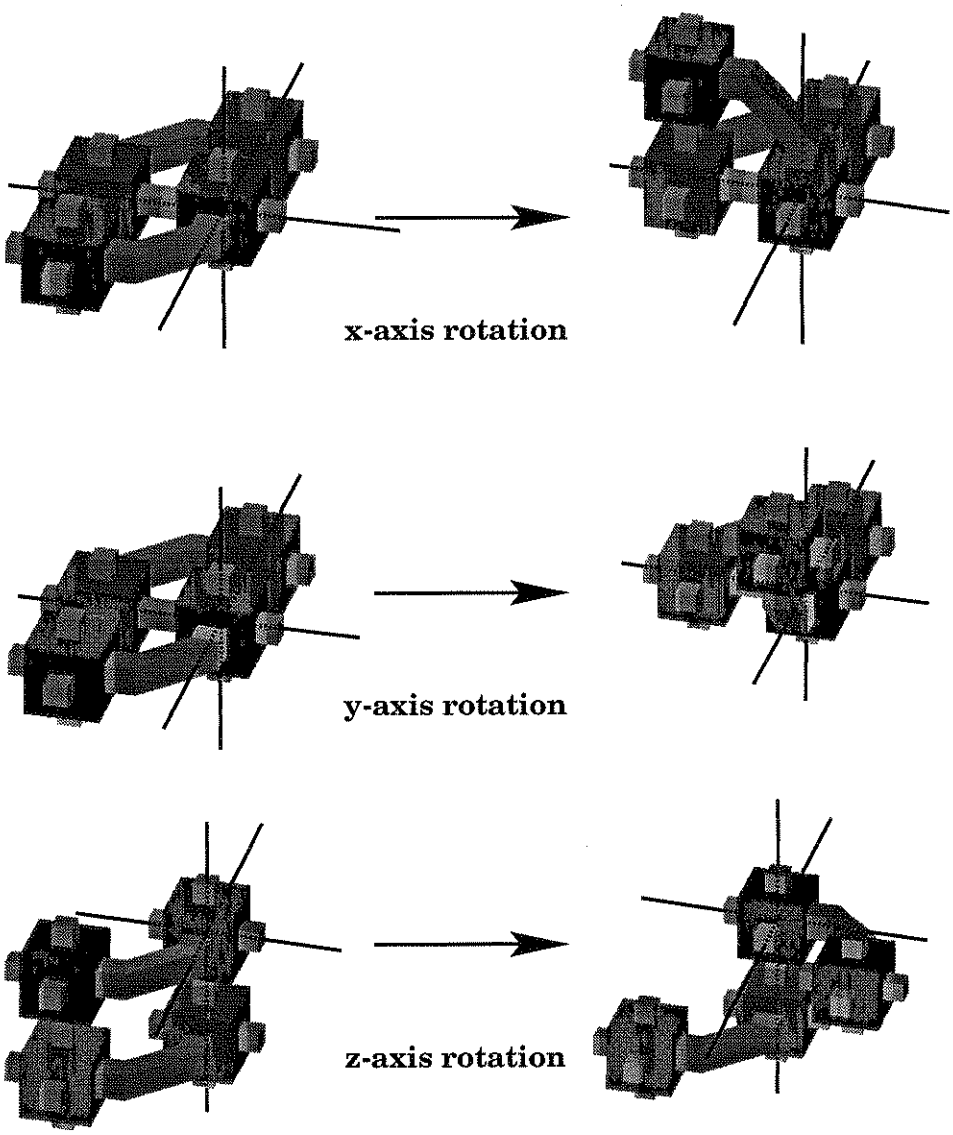

Figure 4: The Molecule Robot's three basic rotations. One endpoint (i.e. "atom") of the moving robot remains stationary, while the other atom and the bond rotate around it. From left to right, the robot rotates 90° relative to the stationary atom's x-axis, y-axis, and z-axis.

We would like to describe transitions for the K_0 class that would represent these x, y, and z-axis rotations. Recall from Definition 2.5 that a transition is a source position, a destination position, and a set of preconditions that must be true for a module to change its position from the source to the destination.

Recall also that a transition's preconditions must be expressed as two sets of positions — a set of positions that must be full, and a set of positions that must be empty. In the K_0 class, each transition requires one position to be full – the actuating position – and a set of empty positions – the collision set.

We can now define twelve disjoint sets of transitions on the L_0 lattice. The set of transitions of the K_0 class is the union of these twelve sets:

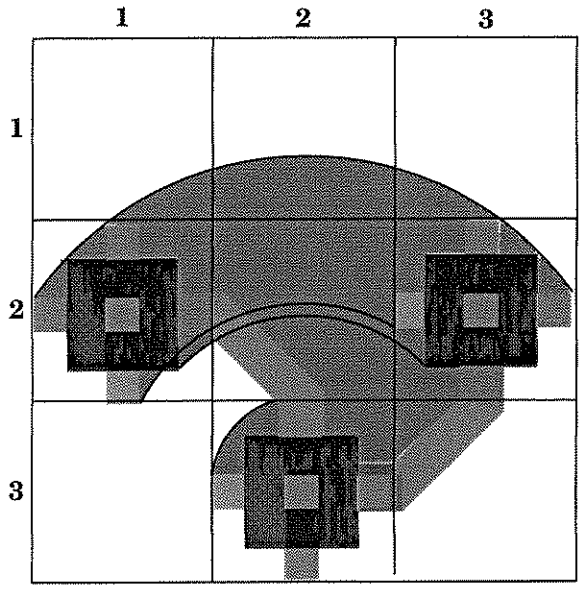

Figure 5: The path of a Molecule Robot as it rotates about the z-axis of on of its atoms. Rotations which would result in collisions with other modules are disallowed.

$$
\begin{aligned}
T_{(i_x,i_y,i_z,i_r)} = \{\ & \{s,d,E,F\}: \\
& (i_x,i_y,i_z,i_r) \in D_{(0,0,0,R_1)}, \\
& s=(x,y,z,r), \text{ where } x,y,z \in \mathcal{Z} \text{ and } r \in \{R_1 ... R_{12}\}, \\
& d=(x+i_x, y+i_y, z+i_z, r+i_r-R_1), \\
& E=\{(x+j_x, y+j_y, z+j_z, r+j_r-R_1) : (j_x,j_y,j_z,j_r) \in E_{0,0,0,R_1}[(i_x,i_y,i_z,i_r)\}, \\
& F=\{s,f\}, \\
& f=(x+k_x, y+k_y, z+k_z, r+k_r-R_1), \text{ where } (k_x,k_y,k_z,k_r) \in F_{0,0,0,R_1}[(i_x,i_y,i_z,i_r) \\
& \}
\end{aligned}
$$

We can now define the set of transitions of the K_0 class:

$$T_0 = \bigcup_{d \in D_{(0,0,0,R_1)}} T_d$$

4 The Motion Planning Problem

In order to reconfigure a system, we need to find a sequence of transitions that can be applied to the starting configuration to transform it into the desired configuration. Recall from Definition 2.5 that in order to apply a transition to a configuration, all of the transition's preconditions must be met by that configuration, and the connectivity requirement must be met.

So, we say that a transition, $T = s, d, E, F$, is *available* in a configuration, C, if C meets all of T's preconditions, and the removal from C of the module at position s would result in a connected configuration.

So, planning for a reconfiguration proceeds by selecting and applying one available transition. After the transition has been performed, the configuration is re-evaluated, and a new transition is selected. This process continues until the desired configuration is achieved.

The time required to plan a reconfiguration using blind search is an exponential function of the number of available transitions. This number is proportional to the *perimeter* [ChPa] of the configuration. In a 3-dimensional regular lattice, the perimeter of a configuration is at best $\Theta(n^{2/3})$ and at worst $\Theta(n)$, where n is the number of positions in the configuration. So, the time to plan a reconfiguration using blind search is an exponential function of n.

We are interested in finding self-reconfiguration planning algorithms that are not subject to exponential time bounds.

5 A Reduction For Motion Planning

Planning algorithms for a type of self-reconfiguring system known as a Metamorphic Robot [PaChSCh] can run in polynomial time. We would like to leverage these planners by designing algorithms that allow a constant number of Molecule Robots to collectively behave as a single Metamorphic Robot.

The rest of this section proceeds as follows. We describe a hierarchy of three modules, where the base of the hierarchy is the Molecule Robot. Each of the latter two modules is composed of multiple copies of the module at the previous level in the hierarchy. We present algorithms that allow these modules to "move" by separating into their underlying components, moving the underlying components, and then re-assembling in a new position. We use these algorithms to define a set of transitions for the new modules. With each level in the hierarchy, the new transitions add flexibility to the self-reconfiguring system.

We show how Molecule Robots can be aggregated to construct a new module called a Double-Pair. We then show how Double-Pair modules can be aggregated to construct an Axes module. Finally, we show that systems composed of Axes modules satisfy the requirements of the polynomial-time planning algorithms described in [ChPa]. Figure 6 illustrates the path of this reduction.

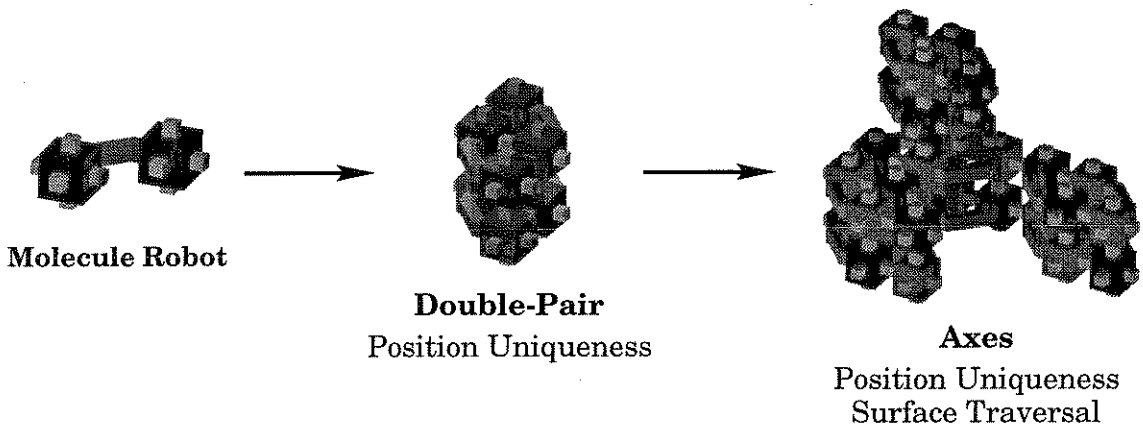

Figure 6: The path of the reduction from Molecule Robots, to Double-Pair Modules, to Axes Modules. Each step increases the flexibility of the transitions in the system.

5.1 The Double-Pair Module and the Class K_1

We are looking to compose a module class that satisfies the position uniqueness and surface traversal properties. We will do this one step at a time, by first composing a class with unique positions that does not satisfy surface traversal. Later, in Section 5.2, we will use this class to compose a class that has both properties.

Let K_1 be a class of module that conforms to a 3D rectangular lattice, and can make straight and diagonal transitions according to certain preconditions. We define this class below:

Let $K_1 = \{L_1, T_1\}$ be a class of modules.
Let $L_1 = \{P_1, I_1, A_1\}$ be a 3D rectangular lattice. i.e.

- $P_1 = \{a \cdot (1,0,0) + b \cdot (0,1,0) + c \cdot (0,0,1) : a, b, c \in \mathcal{Z}\}$
- $I_1 = \emptyset$
- $A_1 = \{((x,y,z) \in P_1, (x \pm 1, y, z))\} \cup \{((x,y,z) \in P_1, (x, y \pm 1, z))\} \cup \{((x,y,z) \in P_1, (x, y, z \pm 1))\}$

Let $T_1 = T_{1S} \cup T_{1D}$ be a set of straight and diagonal transtions where:

- $T_{1S} = \{t = \{p, p + v_1, E, F\} :$

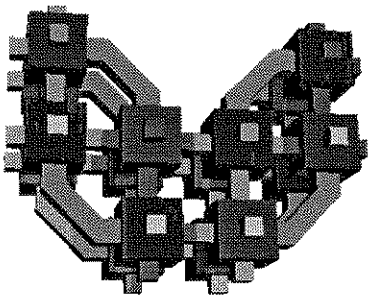

Figure 7: Two adjacent Double-Pair modules. The module on the left is "right-side-up", while the module on the right is "up-side-down". The Double-Pair module is composed of 4 Molecule Robots, and is a member of class K_1.

$$
\begin{aligned}
& E = \{p + v_1, p + v_3, p + v_1 + v_3\}, \\
& F = \{p, p + v_2, p + v_1 + v_2\}, \\
& v_1 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}, \\
& v_2 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\} - \{\pm v_1\}, \\
& v_3 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\} - \{\pm v_1, \pm v_2\} \\
& \}
\end{aligned}
$$

- $T_{1D} = \{t = \{p, p + v_1 + v_2, E, F\} :$

$$
\begin{aligned}
& E = \{p + v_1 + v_2, p + v_2, p + v_3, p + v_1 + v_3, p + v_2 + v_3, p + v_1 + v_2 + v_3\}, \\
& F = \{p, p + v_1\}, \\
& v_1 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}, \\
& v_2 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\} - \{v_1, -v_1\}, \\
& v_3 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\} - \{v_1, v_2, -v_1, -v_2\} \\
& \}
\end{aligned}
$$

Since the set of intersections in the K_1 lattice is null, it is clear that K_1 satisfies position uniqueness. However, as described above, K_1 transitions require empty positions in a direction orthogonal to the direction of motion (i.e. v_3), the class fails to satisfy the surface traversal property. For example, if two surfaces are separated by only one position, a K_1 module would be unable to squeeze between the two surfaces to traverse them.

Lemma 1 *Four Molecule Robots can be aggregated to implement a module of class K_1.*

Let a Double-Pair module be the configuration of 4 Molecule Robots, as shown in Figure 7. 20 inter-module connectors belonging to the underlying Molecule Robots lie on the surface of the Double-Pair module. The Double-Pair module can use these connectors of the underlying Molecule Robots to connect to Double-Pair modules on each of its six faces.

We therefore define a 3D rectangular lattice where each cell in the lattice will exactly fit a Double-Pair module. Imagine that this lattice is colored red and black like a 3D checker board. If Double-Pair modules in black boxes are always "right-side-up", and modules in red boxes are always "up-side-down" (see Figure 7), then any two adjacent Double-Pair modules can connect across at least two inter-Molecule connection points. So, Double-Pair modules have 3D rectangular geometry, and can connect to other Double-Pair modules across all six faces.

Now, we need to show that Double-Pair modules can execute the K_1 transitions described above.

There are 6 straight directions in which a module of class K_1 can move. This direction is described in the class definition (above) by the vector v_1. To satisfy the preconditions, we must also select vectors v_2 and v_3. For these we have 4 choices and 2 choices, respectively. So, there are a total of 48 straight transitions in K_1 that must

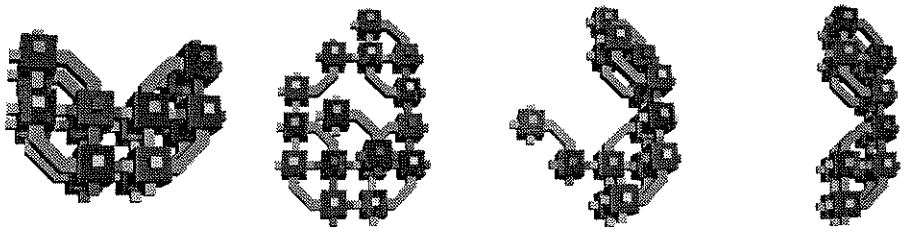

Figure 8: A K_1 diagonal transition in the $x-y$ plane, implemented with Molecule Robots as component modules. The transition requires 64 K_0 transitions and is not necessarily optimal.

be implemented by the Double-Pair module. Due to symmetries in the Double-Pair module, only 12 of these transitions are distinct.

Similarly, to define a diagonal K_1 transition, we must select vectors v_1, v_2, and v_3. Having selected v_1 from 6 choices, there are 4 choices remaining for v_2, and will be 2 choices for v_1. So, there are 48 diagonal transitions that must be implemented. Due to symmetries in the Double-Pair module, only 6 of these are distinct.

To show that a Double-Pair module can implement a particular transition, we must show that in *any* case where that transition's preconditions are satisfied, the Molecule Robots that underlie the Double-Pair module will be able to reconfigure in such a way as to move the Double-Pair module to its new position.

We have discovered sequences of K_0 transitions that demonstrate that Double-Pair modules can implement the 18 distinct K_1 transitions. A sequence implementing one of these transitions must never cause modules to collide, and must never cause the configuration to become disconnected. In order to show that the sequences are valid in any situation where the K_1 preconditions are true, each of these sequences conforms to the following restrictions:

- **Space Restriction** – Let U be the union of the set of positions required to be full and the set of positions required to be empty by the K_1 transition's preconditions. Let $U\prime$ be the set of corresponding positions in the underlying lattice of the K_0 class. Let V be the union of all positions specified in the preconditions of all K_0 transitions in the sequence. V must be a subset of $U\prime$. This is required because positions that are not specified in the preconditions could be either full or empty; their value is not known.

- **Internal Connectivity** – The modules of class K_1 that are specified in the preconditions must remain connected to one another at all times. This ensures that we do not rely on positions outside of what are specified in the preconditions (i.e. positions that may not be full) to keep the internal configuration connected.

- **External Connectivity** – Any double-pair module that is specified in the preconditions, *except* for the moving module, must keep at least one Molecule Robot connected to any neighbor positions that are not specified in the preconditions. While the preconditions guarantee that the removal of the moving module will not disconnect the configuration, they make no similar promises for helper modules. This guarantees that if a helper module is connecting the configuration, it will continue to do so throughout the transition.[2]

We have verified through Prolog simulation that the sequences we have discovered obey transition preconditions and the above restrictions.

5.2 The Axes Module and the K_2 Class

Now, we consider a class of modules that satisfies both the position uniqueness property and the surface traversal property.

[2]It is possible to remove any one of these restrictions by considering the case where a position is full and the case where it is empty. Different sequences can be developed for each case. In Section 5.2, we successfully use this technique to relax the restrictions for one position. However, to remove restrictions on k positions, one would need to consider 2^k cases.

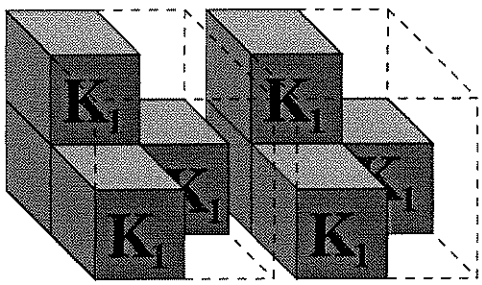

Figure 9: Two adjacent Axes modules. Each Axes module is composed of four modules of class K_1, and is a member of class K_2. In the above Figure, the x-axis of the left module can connect to the origin of the module on the right. Any two Axes modules in adjacent positions can connect in this fashion.

Let K_2 be a class of modules on a 3D rectangular lattice that can make straight and diagonal transitions according to certain preconditions. We define this class below.

Let $K_2 = \{L_2, T_2\}$ be a class of modules.
Let $L_2 = \{P_2, I_2, A_2\}$ be a 3D rectangular lattice. i.e.

- $P_2 = \{a \cdot (1,0,0) + b \cdot (0,1,0) + c \cdot (0,0,1) : a, b, c \in \mathcal{Z}\}$
- $I_2 = \emptyset$
- $A_2 = \{((x,y,z) \in P_2, (x \pm 1, y, z))\} \cup \{((x,y,z) \in P_2, (x, y \pm 1, z))\} \cup \{((x,y,z) \in P_2, (x, y, z \pm 1))\}$

Let $T_2 = T_{2S} \cup T_{2D}$ be a set of straight and diagonal transtions where:

- $T_{2S} = \{t = \{p, p + v_1, E, F\} :$
  $E = \{p + v_1\}$,
  $F = \{p, p + v_2, p + v_1 + v_2\}$,
  $v_1 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$,
  $v_2 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\} - \{\pm v_1\}$
  $\}$
- $T_{1D} = \{t = \{p, p + v_1 + v_2, E, F\} :$
  $E = \{p + v_1 + v_2, p + v_2\}$,
  $F = \{p, p + v_1\}$,
  $v_1 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$,
  $v_2 \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\} - \{v_1, -v_1\}$
  $\}$

Lemma 2 (composite-lemma) *Four Modules of class K_1 can be aggregated to form a module of class K_2.*

Let an Axes module be a configuration of four modules of class K_1, as shown in Figure 9.

Call the K_1 module that is adjacent to three others the Origin. Call the remaining components the X-axis, Y-axis, and Z-axis, respectively. Let m_1 and m_2 be two Axes modules in adjacent positions of $L\prime$. The Origin of one module must be adjacent to either the X-axis, the Y-axis, or the Z-axis of the other module. So, since any module of class K_1 can attach to any adjacent module of class K_1, any two adjacent Axes modules must also be able to attach.

Now, we need to show that Axes modules can reconfigure in such a way as to execute the K_2 transitions described above. There are 24 straight reconfigurations and 24 diagonal reconfigurations that must be demonstrated. However, all of the straight reconfigurations are symmetric, and all of the diagonal transitions are symmetric. So, only two distinct sequences of K_1 transitions are required.

Two such sequences are shown in Figures 10, 11, and 12. Note that two cases are shown for the diagonal transition. In one case, we assume that there is a K_2 module to the left of the moving module. In the other case, we assume there is no such module.

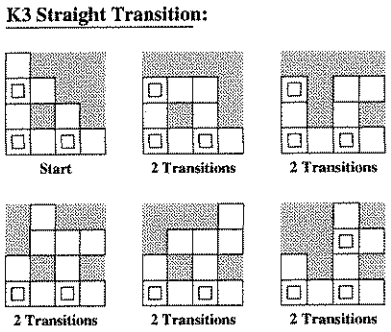

Figure 10: A straight transition of a module of class K_2, implemented with component modules of class K_1. Each frame depicts two layers of K_1 modules: small squares represent modules in the top level, and large squares represent modules in the bottom level.

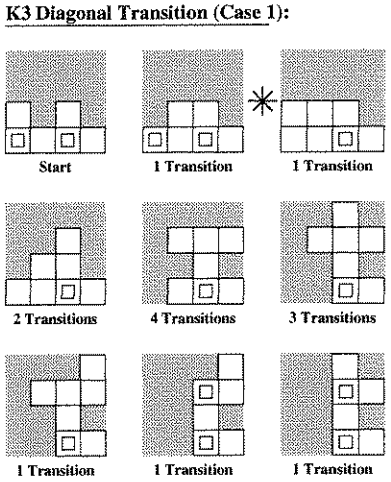

Figure 11: Case 1 of a K_2 diagonal transition, implemented with component modules of class K_1. In this case, we assume that there is no module immediately to the left of the moving module. This assumption is useful during the transition marked with an asterisk (*), where a component module requires open space to the left in order to make a diagonal transition.

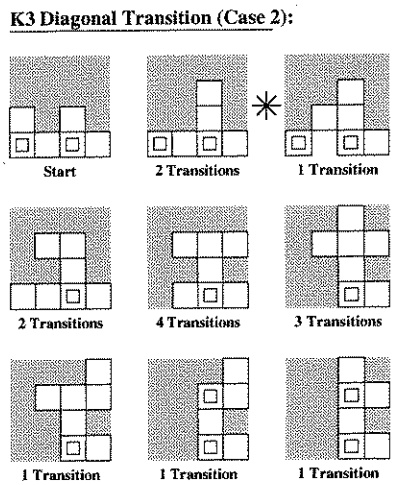

Figure 12: Case 2 of a K_2 diagonal transition, implemented with component modules of class K_1. In this case, we assume that there is another K_2 module immediately to the left of the moving module. This assumption is useful during the transition marked with an asterisk (*), where the component modules separate into two connected sets. The overall configuration does not become disconnected, since precondition 1 guarantees that the module to the left of the moving module must be connected to the remainder of the configuration even if the moving module is removed.

Since Axes modules occupy a 3-dimensional rectangular lattice, can attach to any adjacent Axes module, and can make straight and diagonal transitions provided the preconditions of K_2 are met, we know that we are able to implement a module of class K_2 with 4 modules of class K_1. As discussed in [ChPa], modules that can make such transitions can traverse any continuous surface of a configuration.

Since 4 Molecule Robots can aggregate to a module of class K_1, and since 4 modules of class K_1 can aggregate to a module of class K_2, we can transitively construct a module of class K_2 from 16 Molecule Robots. Since systems of modules of class K_2 satisfy both Position Uniqueness and Surface Traversal, we can use Molecule Robots to construct a system whose motion planning can be performed using the polynomial-time algorithm of [ChPa].

6 Discussion

We have shown how Molecule Robots can be used as building blocks to construct polynomial-time self-reconfiguring robotic systems in three dimensions. The system we constructed uses 16 Molecule Robots per Axes Module. We know by reduction to [ChPa] that the number of Axes Module transitions required to reconfigure our system is $\Theta(n^2)$, where n is the number of Axes modules. Since Axes transitionsrequire a constant number of Molecule Robot transitions, the number of required Molecule Robot transitions is also $\Theta(n^2)$.

References

[ChBu] I. Chen and J. Burdick, Enumerating the Non-Isomorphic Assembly Configurations of a Modular Robotic System, to appear in the *International Journal of Robotics Research.*

[ChPa] G. Chirikjian, Amit Pamecha Bounds for Metamorphic Robots.

[CLBD] R. Cohen, M. Lipton, M. Dai, and B. Benhabib, Conceptual design of a modular robot, *Journal of Mechanical Design*, March 1992, pp. 117-125.

[FuKa] T. Fukuda and Y. Kawauchi, Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator, in *Proceedings of the 1990 IEEE Conference on Robotics and Automation*, pp. 662-667.

[GoOro] Jacob E. Goodman and Joseph O'Rourke Handbook of Discrete and Computational Geometry CRC Press, 1997

[HaSa] G. Hamlin and A. Sanderson, Tetrabot modular robotics: prototype and experiments, in *Proceedings of the IEEE/RSJ International Symposium of Robotics Research*, pp 390-395, Osaka, Japan, 1996.

[KoRu] K. Kotay, D. Rus Motion Synthesis for the Self-Reconfiguring Molecule

[KRVM] K. Kotay, D. Rus, M. Vona, C. McGray The Self-reconfiguring Robotic Molecule.

[McRu] C. McGray, D. Rus Self-Reconfigurable Molecule Robots as 3D Metamorphic Robots

[MKuKo] S. Murata, H. Kurokawa, and Shigeru Kokaji, Self-assembling machine, in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, 1994.

[NeSa] B. Neville and A. Sanderson, Tetrabot family tree: modular synthesis of kinematic structures for parallel robotics, in *Proceedings of the IEEE/RSJ International Symposium of Robotics Research*, pp 382-390, Osaka, Japan, 1996.

[PaChSCh] A. Pamecha, C-J. Chiang, D. Stein, and G. Chirikjian, Design and implementation of metamorphic robots, in *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, Irvine, CA 1996.

[PaKh95] C. Paredis and P. Khosla, Design of Modular Fault Tolerant Manipulators, in *The First Workshop on the Algorithmic Foundations of Robotics*, eds. K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, pp 371-383, 19 95.

[PaKh93] C. Paredis and P. Khosla, Kinematic Design of Serial Link Manipulators from Task Specifications, in *International Journal of Robotic Research*, Vol. 12, No. 3, pp 274–287, 1993.

[Yim] M. Yim, A reconfigurable modular robot with multiple modes of locomotion, in *Proceedings of the 1993 JSME Conference on Advanced Mechatronics*, Tokyo, Japan 1993.

[YMTKuKo] E. Yoshida, S. Murata, K. Tomita, H. Kurokawa, and S. Kokaji, Distributed Formation Control of a Modular Mechanical System, in *Proceedings of the 1997 International Conference on Intelligent Robots and Systems*, Grenoble, France, 1997.