Chapter 4
Lattice-Based Modular Self-Reconfigurable Systems

Kohji Tomita, Haruhisa Kurokawa, Eiichi Yoshida, Akiya Kamimura, Satoshi Murata and Shigeru Kokaji

Abstract This chapter presents a review of research related to self-reconfigurable systems at AIST, particularly addressing their lattice nature. Mainly, three systems are described: Fractum in a 2D hexagonal lattice, and 3D units and M-TRAN in a cubic lattice. Each has distinctive characteristics. Their basic design, reconfiguration methods, and physical implementation issues are discussed and compared.

4.1 Introduction

Typical conventional robots and mechanical systems comprise components of various kinds such as structural elements, mechanisms, and actuators. Components are arranged carefully. The mechanical connectivity among the components does not change. When a robot's task is fixed or its environment is known well in advance, such a design is crucial for realizing high precision, durability, and efficiency in terms of time, space, and energy.

K. Tomita (✉) · H. Kurokawa · E. Yoshida · A. Kamimura
National Institute of Advanced Industrial Science and Technology (AIST),
Tsukuba, Japan
e-mail: k.tomita@aist.go.jp

H. Kurokawa
e-mail: kurokawa-h@aist.go.jp

E. Yoshida
e-mail: e.yoshida@aist.go.jp

A. Kamimura
e-mail: kamimura.a@aist.go.jp

S. Murata
Department of Bioengineering and Robotics, Tohoku University, Sendai, Japan
e-mail: murata@molbot.mech.tohoku.ac.jp

S. Kokaji
Ibaraki, Japan
e-mail: s_kokaji@mail1.accsnet.ne.jp

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_4

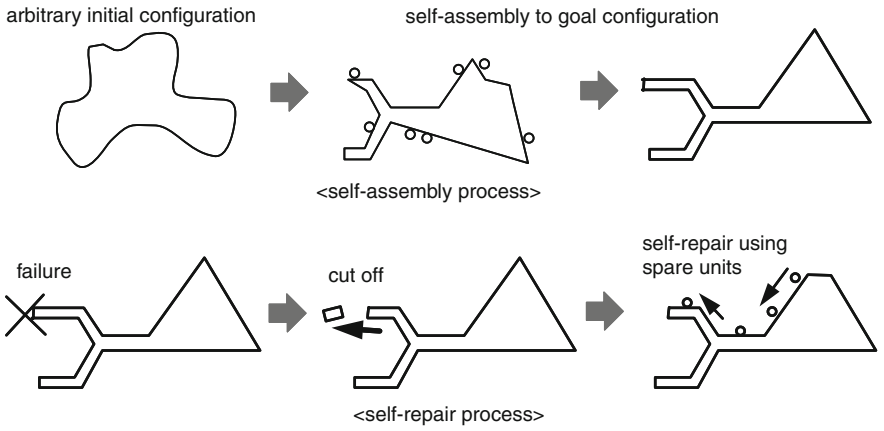


Fig. 4.1 Self-assembly and self-repair [1]

We have been working on robotic systems comprising many identical components called *units*, similar to living systems with their constituent cells and identical genetic information. The units are assumed to have the ability to change their mutual connections independently. Such systems, called modular self-reconfigurable systems, have the following potential benefits.

- Adaptability: Depending on the change of its environment or task, the system can adapt its configuration and function.
- Scalability: Depending on the number of units, different structure and function can be realized.
- Redundancy: Malfunctioning units can be replaced.
- Reliability: The system can continue working in spite of component failures.

Modular self-reconfigurable systems are well suited for self-assembly and self-repair, which is difficult to realize using conventional robots. A process of self-assembly and self-repair is presented in Fig. 4.1. The system assembles itself into a target configuration and achieves its function. If some part is damaged, the system repairs itself and recovers its functionality by cutting off the damaged parts and reassembling itself again using undamaged spare units.

For designing modular self-reconfigurable systems, it is helpful to make a lattice constraint so that the rest positions of the units are restricted to the grid points of a lattice. The translational and rotational symmetry and the regularity of lattices reduce the complexity of modular self-reconfigurable systems both computationally and mechanically. Computationally, it is important that each unit has at most a certain fixed number of neighboring units. For instance, by assigning a discrete state to each unit and by providing state transition rules described in a uniform manner, the global state or configuration of the system can be controlled as in cellular automata [2, 3]. Mechanically, restricting the rest positions of the units to grid points reduces complexity in the design of motions of the units. For instance, global reconfiguration

can be achieved by repetitive local reconfiguration such that a unit on a grid point moves to an adjacent vacant grid point. Such local reconfiguration is much simpler because the possible movements are few.

In spite of the theoretical simplicity, their hardware implementation brings much difficulty. Numerous lattice-based modular self-reconfigurable robots have been proposed [1, 4–6]. Many have only been implemented in a few hardware units or only in simulation. This chapter presents a review of our three lattice-based hardware systems: Fractum, 3D units, and M-TRAN. Among these three systems, the former two were intended mainly to simulate an ideal lattice-based system. M-TRAN was intended to function by itself as a real machine or a robot capable of robotic motion by continuous actuation. This survey presents a review of their basic designs, reconfiguration algorithms, and hardware implementation, and discussion of general problems of lattice-based mechanical systems.

4.2 Fractum

The Fractum is our early prototype system based on the 2D hexagonal lattice [7–9]. Using this system, we developed several algorithms for self-assembly and self-repair. Then we conducted experiments to demonstrate its feasibility.

4.2.1 Basic Design

Figure 4.2 is a schematic view of a Fractum unit. No movable part exists in any single unit. Self-reconfiguration is done by controlling magnetic force.

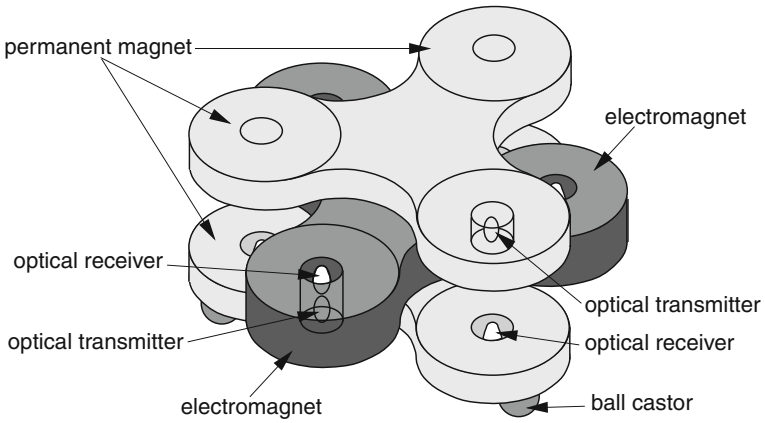


Fig. 4.2 Schematic view of Fractum [1]

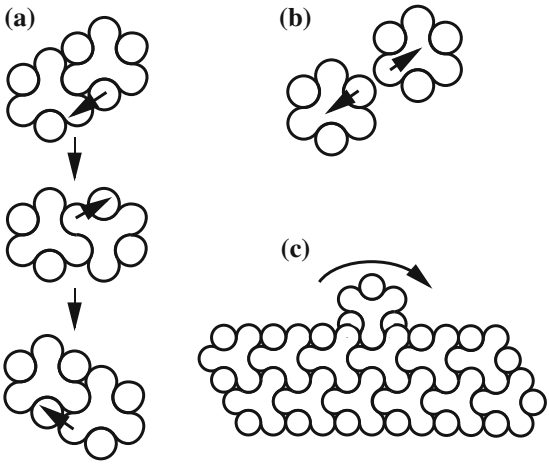


Fig. 4.3 Basic functions of Fractum. **a** Change connection, **b** Cut connection, **c** Transportation [1]

A Fractum unit has a three-layer structure. Permanent magnets are placed in the round parts of the top and bottom layers. Electromagnets are placed in the round parts of the middle layer. The round parts are called arms. By attractive or repulsive force between an electromagnet of a unit and permanent magnets of another unit, the arm of the former unit is attracted to or repelled from the arm of the latter unit. Using this, as shown in Fig. 4.3, changing the polarity of electromagnets realizes (a) change of connective relation between neighbor units, (b) connection cut, and (c) transportation of a unit.

When two units are connected, bidirectional communication is possible between them using the optical transmitters and receivers embedded in the arms. It is used for transmitting units' states.

4.2.2 Algorithm I

A reconfiguration method based on the local connective relation is presented. To make the best use of physically distributed characteristics of the units, it is desirable that every unit have the same software and that the overall system be controlled in a distributed manner without global information to the greatest extent possible. We consider describing a global configuration as a collection of local connective relations.

Each Fractum unit has six connecting arms, and their connective relations are classified into the 12 types shown in Fig. 4.4a. Using these types, the global configuration is described. For instance, a triangle in Fig. 4.4b includes three types: o, K and s. Every type o unit has two neighbors with type K. Such a local connective relation is written as 'o (K, K)'. Similarly, units with type K have neighbors with type o, K, K, and s, and units with type s have six neighbors with type s. This configuration is

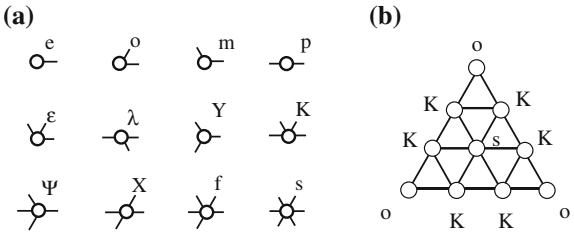


Fig. 4.4 Connection types and triangular configuration [1]

therefore described as

o(K, K),
K(o, K, K, s),
s(K, K, K, K, K, K).

We regard it as a target description.

To achieve a target configuration from an arbitrary initial configuration, each unit is assumed to be given a common target description. Then, depending on the local connective information obtained by communication with the neighbors, each unit repeatedly changes its connection if necessary. The outline of the process is as follows. Each unit calculates its distance of the current local configuration to the given target configuration. If this distance is zero for a unit, then the current local configuration matches some part of target configuration and the unit need not move. If the distance is zero for all units, then the process is regarded as completed and no unit moves. Otherwise, a unit with large distance moves randomly to the right or left.

Hardware experiment of this method using 11 units with the target description (1) is shown in Fig. 4.5.

4.2.3 Algorithm II

Using the method described above, when the structure is large, the system does not always converge to the goal configuration. To improve the success rate and speed of

Fig. 4.5 Reconfiguration experiments using Fractum [1]

convergence, it is effective to provide information not only of the final configuration, but of the intermediate configuration in each construction step.

Figure 4.6 presents an image of the second algorithm. The idea is that a connection network is developed hierarchically from a unit called a kernel, and that undifferentiated units, which are encircling the circumference, are supplied at necessary locations.

Information at each stage is described using the connection types explained above in the form of a lower triangular matrix called a description matrix. From a given target shape, its corresponding description matrix can be generated automatically. Figure 4.7 shows an example of a description matrix and its corresponding configuration at each step. The connection network is generated step-by-step, and the connection type of each unit at each step is described in the matrix. For instance, at stage 4, three units are added, all with type o. They change their types to ε at stage 5, and have type K at stage 6 and later.

When a failure occurs, the system can execute self-repair by cutting off the faulty part and returning to a previous stage. Figure 4.8 shows a simulation of self-assembly and self-repair using this method.

4.2.4 Meta Unit

A group of several units that are arranged in a larger lattice and which have reconfiguration capability as a group in the larger lattice are called meta units. Meta units are intended not only to mimic the original behavior, but also to have more flexibility.

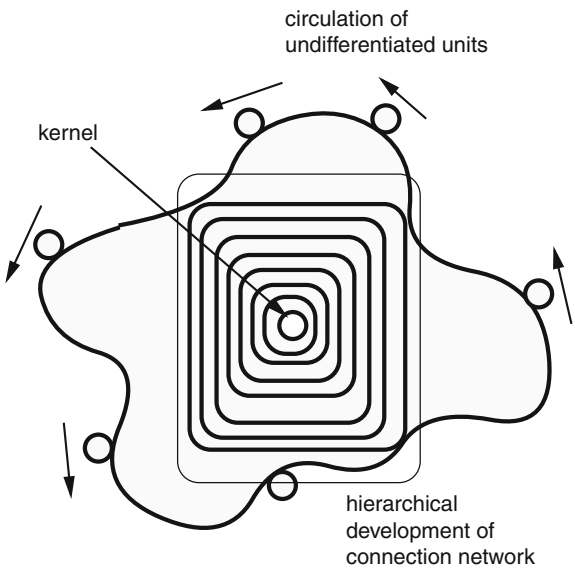


Fig. 4.6 Algorithm II [1]

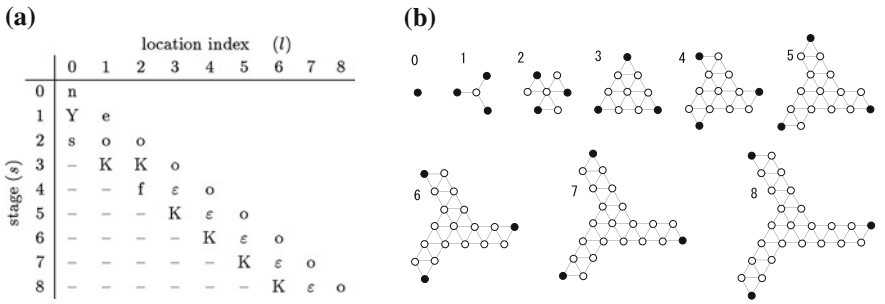


Fig. 4.7 Description matrix and configuration at each stage [1]

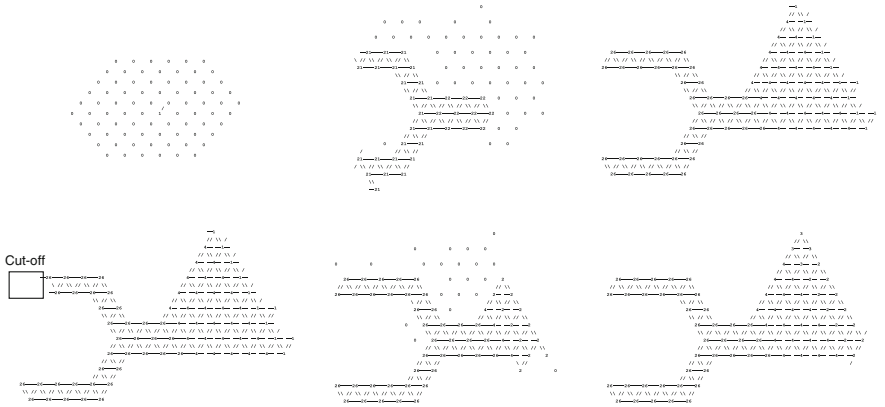


Fig. 4.8 Simulation of self-assembly by Algorithm II

For instance, one can consider a meta unit for Fractum. Using the original units, in the configuration in Fig. 4.9a, it is not possible for the gray unit to move downward and pass through a gap. Using a meta unit composed of 24 original units in Fig. 4.9b, however, by appropriately designing a moving sequence, the gray meta unit can pass through a gap and perform the prohibited motion (Fig. 4.9c).

From a practical perspective, implementing meta units requires many original units and is not so easy. However, when the movability of the original unit is not enough, introducing meta units is effective, as described later with M-TRAN.

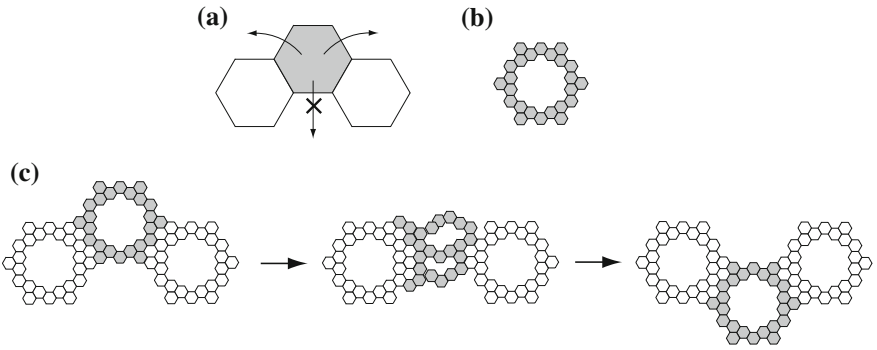


Fig. 4.9 Meta unit for Fractum [1]

4.3 3D Units

Our next system is Three Dimensional Universal Connection System (called as 3D units) [10], which is an extension of 2D hexagonal lattice of Fractum to cubic lattice.

4.3.1 Design

Each 3D unit occupies a grid point in the cubic lattice. Figure 4.10a shows a schematic view of a unit. Each unit has one rotation arm with a connecting hand in every six directions. The hand is for connection and disconnection. The arm is for local reconfiguration by pairwise motion; rotation of itself and its neighbor depending on connective situation. Such rotation enables local reconfiguration as shown in Fig. 4.10b, c. We assume that unit X and unit Y are connected, and that both are connected to their own lower units. Reconfiguration is performed by the following steps:

1. Unit Y and unit Z cut their mutual connection.
2. Unit X rotates the lower arm for 90° around the b axis, resulting in the position change of unit Y.
3. Unit Y and its lower unit connect each other.

Figure 4.11 shows the developed hardware units.

4.3.2 Reconfiguration

A reconfiguration method based on local connection types, which is an extension of the Algorithm I for Fractum, was developed.

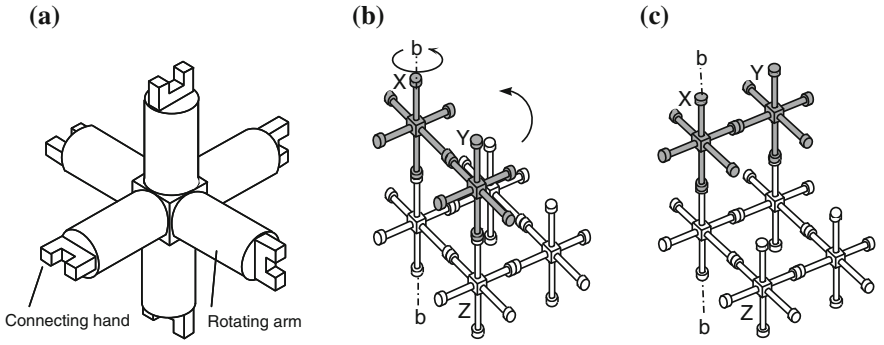
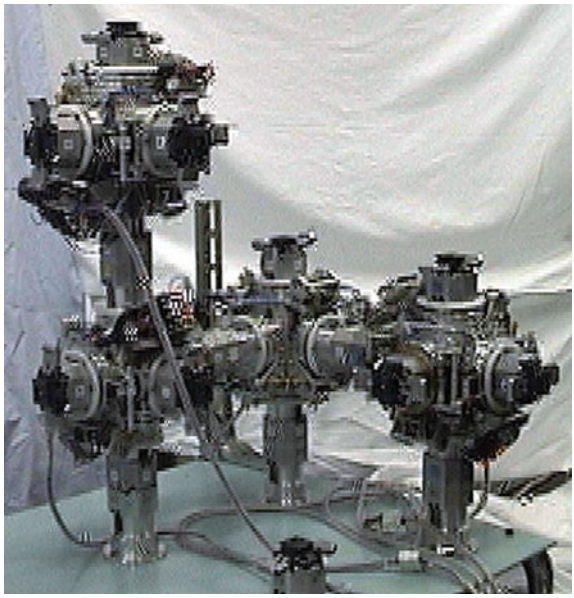


Fig. 4.10 Schematic view of the 3D unit and basic function

Fig. 4.11 3D units hardware



In this system, nine connection types exist, as shown in Fig. 4.12a. Figure 4.12b shows simulation steps using 12 units for constructing a target shape described as

C31(C31, C31, C41),
C41(C31, C31, C41, C41).

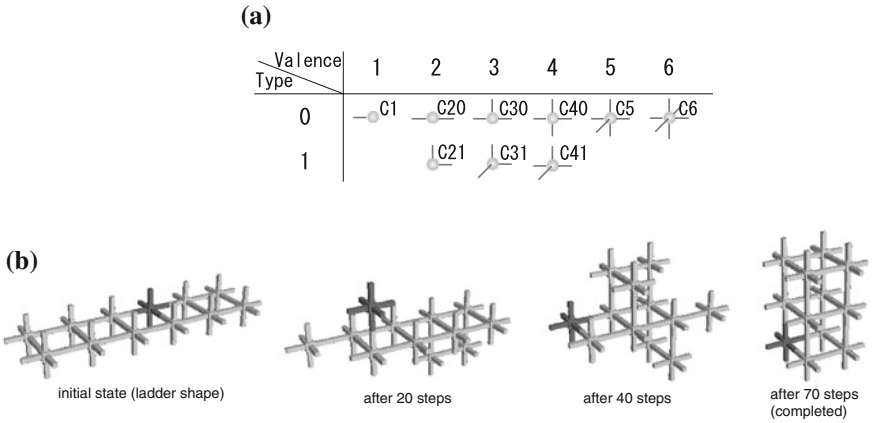


Fig. 4.12 Connection types and reconfiguration simulation of 3D units. **a** Connection types of 3D units, **b** Reconfiguration simulation

4.4 M-TRAN

We developed Modular TRANsformer (M-TRAN) by simplifying the 3D units so that two units are mechanically combined into a module [11–17]. Each module has only two rotational degrees of freedom, but modules have 3D reconfiguration capability as a group.

4.4.1 Design

An M-TRAN module comprises three parts: an active block, a passive block, and a link between them, as in Fig. 4.13. Each block has the shape of a half-cube and half-cylinder, and can rotate 180°. Each block has three flat surfaces for connection. An active block connects only with a passive block of another module. Because of the parity property of the cubic lattice, this polarity is not a restriction.

In contrast to the previous two systems, M-TRAN is largely asymmetric, i.e., possible motions by an M-TRAN module differ depending on its posture. For example, if we assume that a single module is placed on a plane filled with modules as in Fig. 4.14, the module can move by rolling (a) or pivoting (b), depending on its posture. These two postures can be changed mutually with the help of an additional module as in (c). In many reconfiguration sequences of M-TRAN, cooperation of two or more modules in this manner is indispensable.

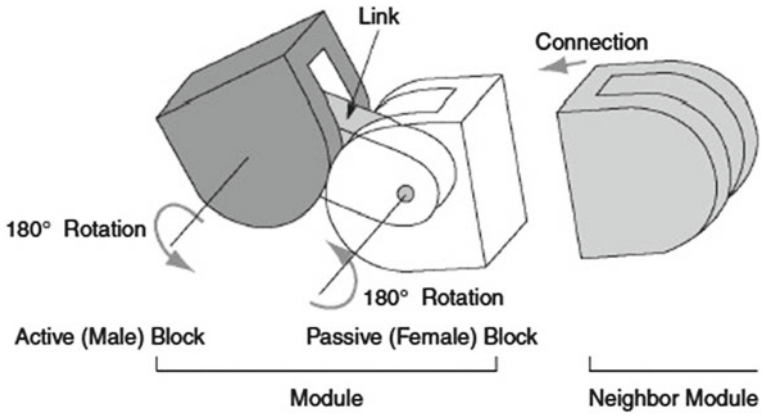


Fig. 4.13 Schematic view of M-TRAN module

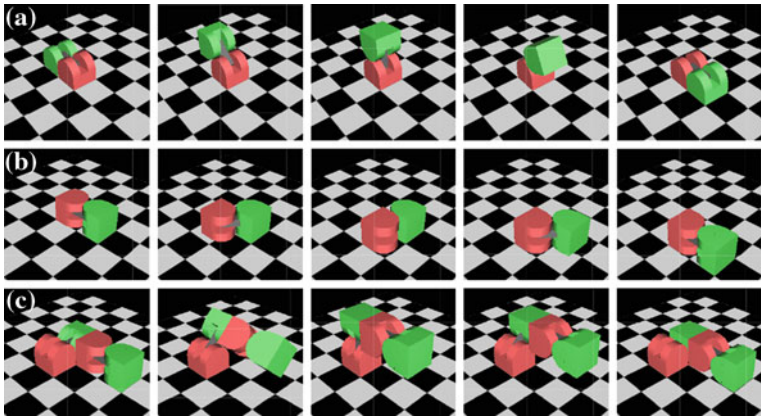


Fig. 4.14 Motions of M-TRAN modules

4.4.2 Reconfiguration

Designing the reconfiguration procedure of M-TRAN modules is not straightforward because the possible motion and connection of a module is restricted by its simplicity and anisotropy. Two neighboring modules in the lattice cannot always be mutually connected because of the limited number of connecting surfaces on each block. A module's motion is also restricted by nearby modules because of collision.

We have manually developed various reconfiguration sequences for small scale reconfiguration up to 10 modules, including transformation of a four-legged structure to a linear structure and its reversal (Fig. 4.15a, e).

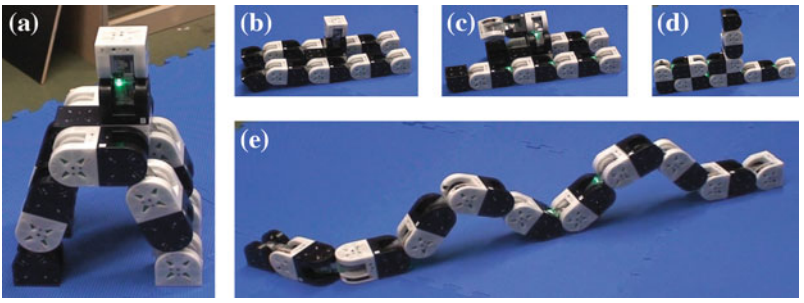


Fig. 4.15 Reconfiguration experiments from a four-legged robot to a linear robot [1]

This small scale reconfiguration is from one initial configuration to another. Therefore, it is different from self-assembly as discussed in the previous sections. To realize self-assembly for a larger reconfiguration, using meta units and introducing regularity to the whole system is beneficial. Herein, we present some regular structures for reconfiguration.

The first is shown in Fig. 4.16a: meta-modules, each of which comprises four modules, are connected linearly. A pair of modules called a converter is attached. This structure is capable of flow motion by the repetitive reconfiguration of transporting the tail meta module to the head position. Turning horizontally or vertically is also possible when assisted by the converter.

The second one (Fig. 4.16b) is a simpler structure for linear flow motion. It comprises two lines of modules. Reconfiguration of this structure was confirmed experimentally.

The next structure is two-dimensional, as shown in Fig. 4.16c. Each meta module comprises four modules. If this structure with sufficient number of modules is placed on a plane, then it can move to a desired direction by flow motion and can take various 2D shapes (Fig. 4.16d).

The final structure is three-dimensional (Fig. 4.17). Each meta module is the same as the previous one, but meta modules are connected in a different way to constitute the cubic lattice. Local reconfiguration procedures for meta modules were developed, but it will be difficult to realize on earth because of gravity.

Individual reconfiguration processes for each step motion require much communication and control including connection and disconnection and rotation, but discussion of the related details is omitted here. By describing them as state transition, the processes can be, in principle, formulated directly as finite state automata.

4.4.3 Robotic Motion

We briefly introduce robotic motion by M-TRAN modules by relaxing the lattice constraint. After reconfiguration into an appropriate structure under a lattice constraint,

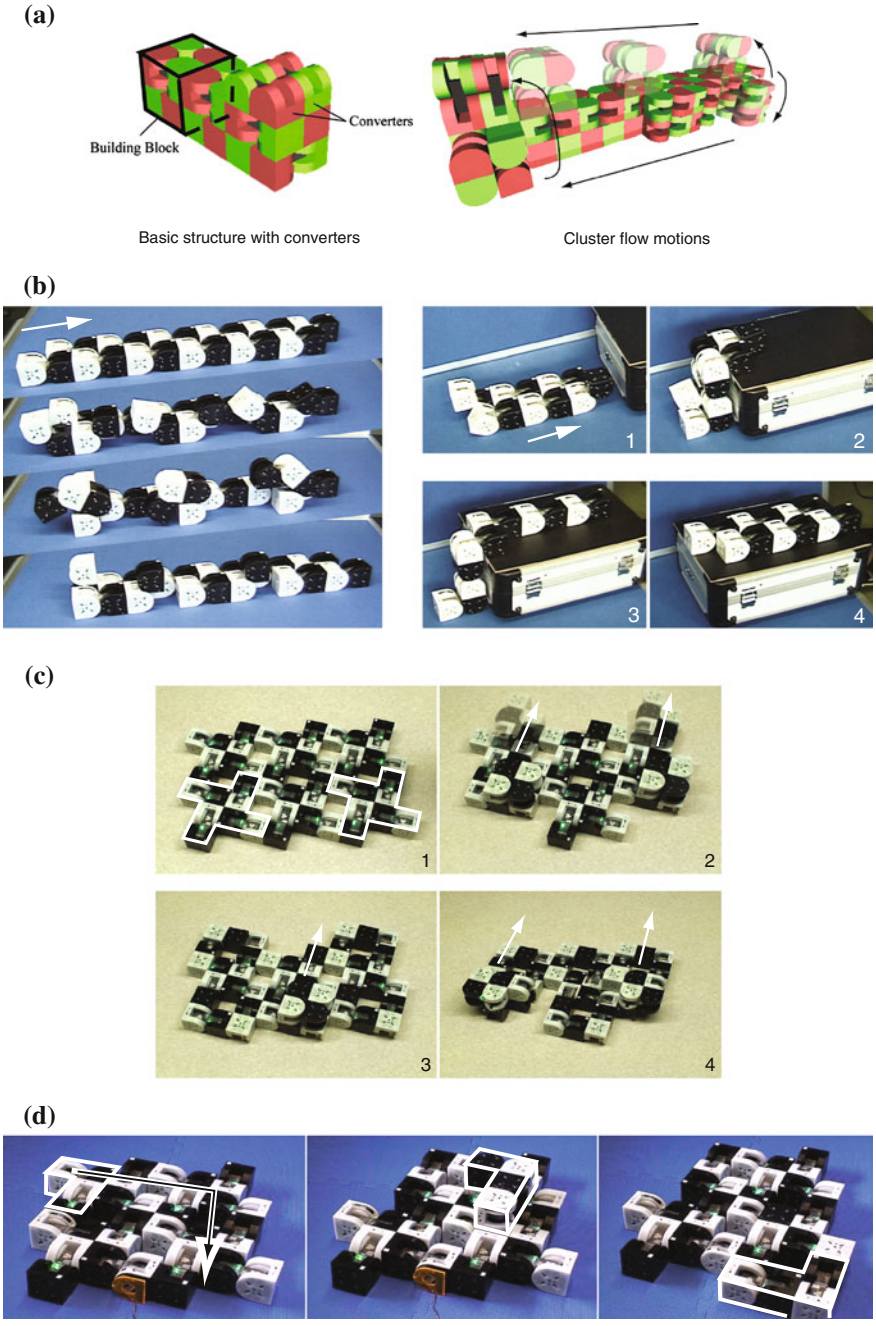


Fig. 4.16 Regular structures of M-TRAN [1]

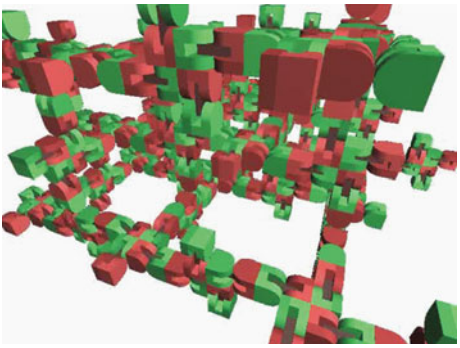


Fig. 4.17 3D regular structure of M-TRAN

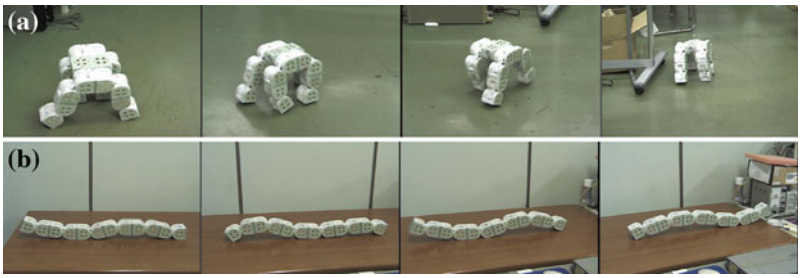


Fig. 4.18 Various locomotion patterns

the modules as a whole can perform robotic motions such as walking. Figure 4.18 shows locomotion experiments of four-legged and linear robots. In fact, locomotion control for each robot is based on a central pattern generator (CPG), the network of which was obtained by a genetic algorithm.

4.5 General Problems of Lattice-Based Mechanical Systems

In this section, we examine hardware issues of general lattice-based systems. Among the three systems described above, the former two, namely Fractum and the 3D units, were intended mainly to simulate ideal lattice-based systems. An ideal system is a distributed physical system comprising many identical units. Each unit is considered simple both in motion and in information processing capabilities and each produces motion according to local rules or local decision making such as a cellular automaton. It can be as small as a molecule or a biological cell, so physical and mechanical realization was not the main objective, and mechanical performance was not an issue. Development of these systems actually left behind many physical problems.

The three generations of M-TRAN, namely M-TRANs I, II, and III, were developed to overcome such problems in mechanical realization. It was aimed to work by itself as a real machine or a robot. Designs of two types were combined, which were designated afterwards as lattice-based and chain-based. Improved performance of the unit was necessary because a chain-based modular robot as a whole must work as a locomotive robot (Fig. 4.18) under rather centralized control.

4.5.1 Improvement in M-TRAN Hardware

An M-TRAN module equips stronger actuators and a more powerful microprocessor than the other two systems. Each cubic block of an M-TRAN module equips only one actuator for motion, so a module needs to carry other modules for reconfiguration. For robotic locomotion, such as walking in a legged configuration (Fig. 4.18), a small number of joint motors must support and move the whole body, so a stronger actuator is necessary. At the same time, joints' motion must be synchronized among modules, and modules need to respond fast to an input to a module's sensor. If dynamic control is sought, more complicated numerical processing is necessary. Therefore, an M-TRAN III module has a fast processor (32 bit CPU, SH-II; Renesas Electronics Corp.) and fast and global inter-module communication (CAN bus).

Each module of M-TRAN II and III has a battery. This not only made the whole body locomotion possible but also helped reconfiguration as a lattice-based system. For Fractum and 3D units, their reconfiguration invariably made tethers for power supply tangle. To address this problem, M-TRAN I was designed so that a single set of tethers is connected at one end of the whole system and power is transmitted to all the units via connection. This method, however, proved to be ineffective because the transmission loss was too large when several units were connected serially.

4.5.2 General, Physical Problem in Modular Reconfigurable Systems

As a lattice-based system, M-TRAN is based on a cubic lattice, but a single M-TRAN module comprises two cubic units. This design presents the important benefits of solving problems encountered by Fractum and the 3D units. Simultaneously, it presents many shortcomings. Each such problem and corresponding design solution relates to others. It is not easy to analyze problems systematically and to derive a design guide. Here, some general and important issues will be explained along with a brief introduction of others [1].

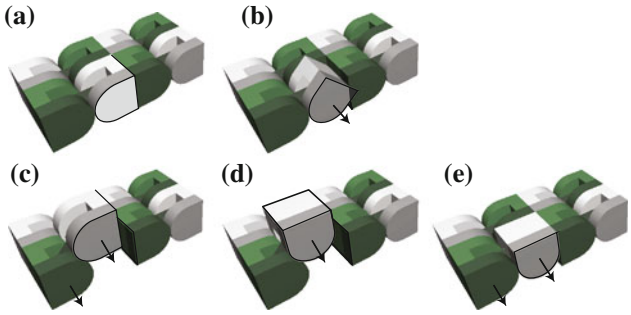


Fig. 4.19 Collision avoidance by parallel axis motion. Although collision is unavoidable by a single axis motion as in **b**, two axis motion avoids it as **c**, **d**, and **e**

4.5.2.1 Unit Shape

With a lattice-based logical system, it is considered that any unit in a cell can move to its adjacent neighbor void cell based on its state, its neighbors' states, and the rules. With a physical system, such motion is not always possible depending on the unit shape, kinematic design, local configuration, and physical performance of actuators. Although a circular or spherical unit can move without colliding with other units, a square or cubic unit cannot always do so, depending on its motion mechanism and surrounding units. However, a circle or a sphere contacts with its neighbor only via a point, which makes it difficult to ensure a rigid connection and strong actuation. A square or a cube is beneficial not only in terms of precision, rigidity of connection, and in actuation strength. It also makes a larger interior space useful for the installation of mechanisms, actuators and circuits. In the case of M-TRAN, with its semi-cubic shape, a problem of collision is partially relaxed by two parallel axes of a single module (Fig. 4.19).

4.5.2.2 Geometric Error, Structure, and Deformation

In contrast to identical units of an ideal lattice-based system, any mechanical product invariably has errors in its geometry. In actual production performed by a precision machine, elements are selected and combined with others so that the error of one element compensates an error of others. Regarding a modular system, all the units must fit at any position. Such compensation is not possible (Fig. 4.20).

With geometric errors, multiple units cannot fit perfectly, so the elasticity of units is necessary to assemble units. In the usual case of assembling a machine or a structure, no element is assembled tightly when it is assembled to others, but all the elements are tightened gradually to distribute stress and strain over the whole body. Such gradual tightening is not considered for a lattice-based machine. Therefore, when units are

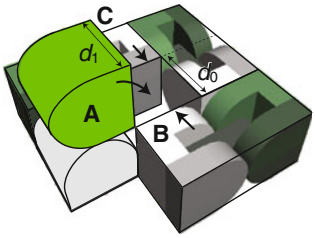


Fig. 4.20 Problem of geometric error. If $d_1 > d_0$, then A cannot move into the gap separating B and C

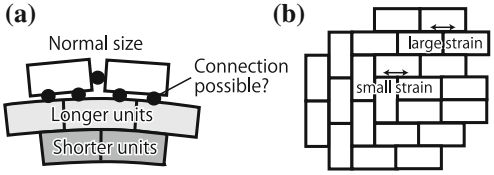


Fig. 4.21 Building a large structure. **a** Irregular units, **b** Non-uniform strain

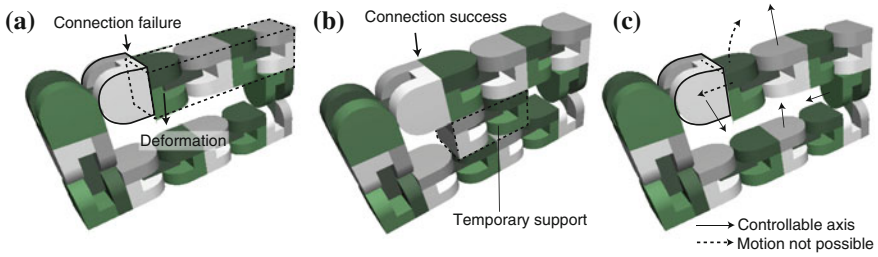


Fig. 4.22 Deformation and control. **a** Gravitational bending of a cantilever beam might cause large misalignment and connection failure. **b** Deformation in **a** is avoided using a supporting module. **c** Without sufficient motion DOFs, deformation in **a** cannot be corrected by control

assembled adding one after another, error will accumulate so that with some number of units, a new unit cannot be added without violating some deformation limit or an actuator's maximum output (Fig. 4.21).

However, elasticity cannot be set too weak. With weak elasticity, deformation of a cantilever beam under gravity might become so large that other units cannot move into the space under the beam or large positioning error might produce an unsuccessful connection (Fig. 4.22a).

Consequently, elasticity and rigidity are required simultaneously. Their proper balance cannot generally be attained using a conventional mechanical system. Regarding M-TRAN, this problem is sometimes avoided by the meticulous design of a reconfiguration sequence. For example, such a vulnerable part as a cantilever beam is

temporarily supported by another module (Fig. 4.22b). This method of avoidance, however, tends to require planning of the whole system. For that reason, a distributed and decentralized nature of a lattice-based system is not always attained.

4.5.2.3 Density

Because an M-TRAN module lacks full symmetry for a cubic unit, it must carry at least one other unit for reconfiguration; its actuator needs to lift its half part plus at least another full module. When an actuator and a battery of sufficient power are given, the weight of a module is roughly determined. With a given weight, and with available force and torque, the smaller the module, the greater the number of modules that can be lifted. Therefore, a higher density of a module is better. Actually, M-TRAN II and III are, although not optimized, quite packed. With its current size, i.e., of about 650 mm cube, it's unlikely that drastic, mechanical improvement in M-TRAN can be made if it is produced using currently available technology.

4.5.2.4 Control

Robotics can be regarded as a science of intelligence in mechanical control. If a module's geometry is not sufficiently precise or if structural deformation is not negligible, then some control or adaptation should be integrated into the system.

Designing a module to have controllability of geometry might make the problem worse, making the module more complicated and heavy, hence more vulnerable to external forces and easier to deform. For a single module, to measure or correct misalignment as in Fig. 4.22a is not easy and mostly impossible. Cooperation of multiple modules is also not a solution, because sensors and actuators of surrounding modules do not always possess sufficient degrees of freedom for feedback control as shown in Fig. 4.22c. Consequently, a straightforward approach to feedback control is not practical, and is, from the very first, not a proper approach to a lattice-based mechanical system, in which an ideal unit is presumably able to move from one cell to another with a switching motion.

For M-TRAN, each actuator is position-controlled mostly at five discrete angles in 45 deg step. There is no position sensor except angle sensors for the servo controllers and a sensor detecting completion of connection with another module. With such a setting, the modules' motions were controlled in a feed-forward manner. Feasible sequences of reconfiguration such as those in Figs. 4.15 and 4.16c were designed and examined in experiments using a trial-and-error process.

4.5.3 Achievement by M-TRAN

In case of small-scale self-reconfiguration, in which up to 10 modules change their configurations among pre-designed robotic ones, M-TRAN III modules realized

many steps of lattice-based transformations with a small failure ratio. In such cases, geometric errors were not severe. Because robotic motions are intended after reconfiguration, synchronized re-tightening of the whole connection was made to relax possible concentrated strain in the whole structure, thereby avoiding the problems described above in Sect. 4.5.2.2.

Self-reconfiguration in a larger structure as shown in Fig. 4.16c, avoided the problems presented in Fig. 4.20 and Fig. 4.21 because the structure is planar and is not fully packed but instead contains spaces. Some trials of reconfiguration experienced connection failures because of the problems explained in Sects. 4.5.2.2 and 4.5.2.4. So various sequences of reconfiguration for the same target configurations were designed and experimented, and better ones were selected.

Truly three-dimensional structures resembling those shown in Fig. 4.16d and their self-reconfiguration were not tried. They seem far from feasible by the current M-TRAN system because of the problems and difficulties described above.

4.6 Conclusions

We have reviewed self-reconfigurable robots particularly addressing their lattice nature. Three systems, Fractum, 3D units, and M-TRAN, were described. Hardware problems were discussed while taking M-TRAN as a typical case.

Although all the systems, based on latest mechanical and electronic technology, have achieved fairly useful results verifying a lattice-based system, assigning too much emphasis to such mechanical and electronic realization might lead to a dead end. Considering discussions presented in the previous section, the feasibility of a lattice-based mechanical system in general is not assured even when several numbers of units are developed and only with them, varieties of lattice-based motions are verified by experimentation. A breakthrough beyond the achievements of research and evaluations of modular robotics conducted in the past will require considerable numbers of new ideas related to design and a new method of manufacturing and working with vast amounts of units in a simple manner and at low cost.

Recent technologies, such as microfabrication and nanofabrication, DNA tiling, and other molecular nanotechnologies, seem promising. For example, DNA nanotechnology can produce molecular-scale components such as structures, logic gates, sensors, and actuators, which, if integrated properly, can realize molecular scale robots. Working in liquid might mitigate most of the problems listed above. Moreover, a chemical power supply can be made to all units without tethers and batteries. Controllable microparticles or microshells are anticipated for various applications such as drug delivery, such that drugs are contained, transported, and released by micro shells.

For the future development of such systems, the study of robotics is no doubt indispensable. Moreover, the fusion of diverse disciplines of science and technologies will be strongly required, such as nanotechnologies, molecular biology, information technology, and systems science including the study of lattice-based systems.

References

1. Murata, S., Kurokawa, H.: Self-Organizing Robots. Springer, Berlin (2012)
2. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Generic decentralized control for lattice-based self-reconfigurable robots. Int. J. Robot. Res. **23**(9), 919–937 (2004)
3. Stoy, K.: Using cellular automata and gradients to control self-reconfiguration. Robot. Autonom. Syst. **54**(2), 135–141 (2006)
4. Murata, S., Kurokawa, H.: Self-reconfigurable robots. IEEE Robot. Autom. Mag. **14**(1), 71–78 (2007)
5. Stoy, K., Brandt, D., Christensen, D.J.: Self-Reconfigurable Robots: An Introduction. MIT Press, Cambridge (2010)
6. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robot. Autom. Mag. **14**(1), 43–52 (2007)
7. Murata, S., Kurokawa, H., Kokaji, S.: Self-assembling machine. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 441–448 (1994)
8. Tomita, K., Murata, S., Kurokawa, H., Yoshida, E., Kokaji, S.: Self-assembly and self-repair method for a distributed mechanical system. IEEE Trans. Robot. Autom. **15**(6), 1035–1045 (1999)
9. Yoshida, E., Murata, S., Tomita, K., Kurokawa, H., Kokaji, S.: An experimental study on a self-repairing modular machine. Robot. Autonom. Syst. **29**(1), 79–89 (1999)
10. Murata, S., Kurokawa, H., Yoshida, E., Tomita, K., Kokaji, S.: A 3-D self-reconfigurable structure. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 432–439 (1998)
11. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic locomotion design and experiments for a modular robotic system. IEEE/ASME Trans. Mechatron. **10**(3), 314–325 (2005)
12. Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T., Murata, S.: Distributed self-reconfiguration of M-TRAN III modular robotic system. Int. J. Robot. Res. **27**(3–4), 373–386 (2008)
13. Kurokawa, H., Yoshida, E., Tomita, K., Kamimura, A., Murata, S., Kokaji, S.: Self-reconfigurable M-TRAN structures and walker generation. Robot. Auton. Syst. **54**(2), 142–149 (2006)
14. Murata, S., Tomita, K., Yoshida, E., Kurokawa, H., Kokaji, S.: Self-reconfigurable robot. In: Procedings of International Conference on Intelligent Autonomous Systems, pp. 911–917 (1999)
15. Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., Kokaji, S.: M-TRAN: self-reconfigurable modular robotic system. IEEE/ASME Trans. Mechatron. **7**(4), 431–441 (2002)
16. Ostergaard, E.H., Tomita, K., Kurokawa, H.: Distributed metamorphosis of regular M-TRAN structures. In: Distributed Autonomous Robotic Systems 6, pp. 169–178. Springer (2007)
17. Yoshida, E., Matura, S., Kamimura, A., Tomita, K., Kurokawa, H., Kokaji, S.: A self-reconfigurable modular robot: reconfiguration planning and experiments. Int. J. Robot. Res. **21**(10–11), 903–915 (2002)