A membrane computing framework for self-reconfigurable robots

Dongyang Bie[1] · Miguel A. Gutiérrez-Naranjo[3] · Jie Zhao[2] · Yanhe Zhu[2]

Abstract

Self-reconfigurable robots are built by modules which can move in relationship to each other, which allows the robot to change its physical form. Finding a sequence of module moves that reconfigures the robot from the initial configuration to the goal configuration is a hard task and many control algorithms have been proposed. In this paper, we present a novel method which combines a cluster-flow locomotion based on cellular automata together with a decentralized local representation of the spatial geometry based on membrane computing ideas. This new approach has been tested with computer simulations and real-world experiments performed with modular self-reconfigurable robots and represents a new point of view with respect other control methods found in the literature.

Keywords Modular robots · Membrane computing · Distributed control · Self-reconfiguration · Cellular automata · P systems

1 Introduction

Self-reconfigurable robots (Fukuda and Nakagawa 1988; Baca et al. 2017) are robots consisting of modules which can move in relationship to each other, which allows them to change the physical form of the whole robot. Such feature allows the robot to optimize their shape for different tasks. According to Stoy et al. (2010), the problem of self-regulation can be settled as follows: *Given an initial configuration and a goal configuration, find a sequence of module moves that will reconfigure the robot from the initial configuration to the goal configuration*. Choosing an appropriate mechanism (Yim et al. 2007) for controlling the self-reconfiguration is an extremely hard task, due to the diversity and various scales of available motion strategies. Current approaches can be generally divided into two categories: centralized control and decentralized control. It has been proved that the centralized control is a NP problem (Hou and Shen 2014) and decentralized approaches are currently the focus in order to achieve effective solutions. Nonetheless, the main problem in decentralized control is how to provide *global* sense about the desired configuration to local modules (Stoy 2015a).

This problem has been studied from different points of view. One of the most interesting is to consider nature as a source of inspiration. In the literature, several bio-inspired methods have been applied for the distributed control of self-reconfigurable robots, among them, we can cite methods based on cellular automata (CA, for short) (Zhu et al. 2015; Wu et al. 2005; Butler et al. 2001b) or particle swarm optimization (Zhao et al. 2015). In this paper, we propose a novel method which combines a cluster-flow locomotion based on CA together with a decentralized local representation of the spatial geometry based on membrane computing ideas. Both the decentralized localization of independent modules and modular level predictions of global state are managed. To the best of our knowledge, this is the first time where membrane

✉ Yanhe Zhu
yhzhu@hit.edu.cn

Dongyang Bie
bdy@nankai.edu.cn

Miguel A. Gutiérrez-Naranjo
magutier@us.es

[1] The Institute of Robotics and Automatic Information Systems, College of Computer and Control Engineering, and the Tianjin Key Laboratory of Intelligent Robotics, Nankai University, Tianjin, China

[2] State Key Laboratory of Robotics and System (HIT), Harbin Institute of Technology, Harbin, Heilongjiang Province, China

[3] Department of Computer Science and Artificial Intelligence, University of Seville, Seville, Spain

computing is used for the development of self-reconfigurable robots.

The proposed method represents a novelty in the framework of self-configurable robots in a double sense, from a theoretical and practical point of view. From a theoretical side, this paper proposes an abstract representation of the robot beyond its physical representation. This abstract representation opens a door for theoretical studies related to formal languages and more complex representation. Moreover, the chosen abstract representation has a biological inspiration and bridges the area of self-configurable robots with membrane computing field. These bridges can allow the flow of ideas, problems and solutions enriching both research areas. From a practical point of view, the proposed solution is based on two of the basic features of one of the most studied membrane computing devices, the so-call cell-like P systems: On the one hand, the tree-like graph structure which can be abstracted from the hierarchical arrangement of vesicles in an eucaryotic cell. Such tree-like structure allows the flow of information between a vesicle and the vesicles placed inside of it and this inspires for establishing a formal representation of the relative position of robotic modules which provides a natural solution to the decentralized localization problem in self-configurable robots (Stoy and Nagpal 2007). On the second hand, in membrane computing the information is encapsulated in vesicles and encoded by multisets of simple objects by following the biological metabolites placed inside the alive cells. The key point for the use of such multisets in the framework of self-configurable robots is the *interpretation* of the objects. As it will be pointed out below, such objects can represent the length or the relative angle of a module of the robot. But the meaning of symbols in such interpretation can be changed according to different targets, which opens new research possibilities.

We would also like to remark the scalability of the proposed solution in the number of modules. The proposed method for self-reconfiguration of modular robots is convergent to target configurations and scalable in the number of used modules. For the sake of simplicity, the presented method will be illustrated with some simple configurations. Obviously, more complex robots need more modules, but the principles of the decentraliced representation are the same. At the end of the paper we will show some studies of such scalability.

In the literature, there are a few absolutely decentralized algorithms without either external controller or predefined configurations (see e.g., Yoshida et al. 1998; Bojinov et al. 2002; Christensen 2006; Gilpin and Rus 2012; Zavlanos and Pappas 2008). The result structure emerges through independent modules interacting with surroundings. In such way, the scalability of those methods has a high degree of dependence on the distributed nature of modular robots and hence, the reconfiguration process can hardly converge to a satisfactory structure.

In order to guarantee convergence of self-reconfiguration process, a global-centralized to local-distributed model has been widely used. With an external controller, target configurations are translated to executable rules for independent modules. The remaining self-reconfiguration depends on the automatic interaction of decentralized modules. In this way, Stoy (2004) and Stoy and Nagpal (2007) take an overlapping bricks approach to convert the goal configurations into lattice automata rules and Rubenstein and Shen (2010) convey a complete description of the desired configuration to each module in the system. More similar research work can be found in Jones and Mataric (2003), Goldstein et al. (2005), Walter et al. (2005), Funiak et al. (2009) and Pillai et al. (2006).

While allowing large collections of independent modules to form determinate shapes, these works have one common problem in the limitation of attached assumptions for decentralized localization. For example, the localization capability of a module to detect whether it is located inside the approximated volume (Fitch et al. 2005; Zoppi 2011) or on the exterior surface (Fitch and McAllister 2010). This is unpractical for mechanical robots and has been a corner stone of self-reconfiguration control for physical robots (Stoy 2015b). In order to deal with this problem, an extended turtle interpretation for decentralized localization of independent modules through distributed local communication is considered in this paper. The extended turtle interpretation can generate module-level predictions about global self-reconfiguration state through interpreting symbols in membrane systems.

Self-reconfiguration by the proposed method takes place in the development style through modules continuously moving to needed areas. According to module-level predictions transforming in the robotic system, individual modules move in parallel. Because of the high frequency of physical interactions between neighboring modules, and the constraints arising from actuator geometry and power limitations, local movements pose a significant challenge for multi-agent control (Bojinov et al. 2002) and hence, local control of decentralized modules is one of the key elements for successful distributed self-reconfiguration.

Finally, we would like to briefly recall some ideas about the bio-inspired computational research areas used in this paper, membrane computing and cellular automata. Membrane computing[1] is an emergent branch of natural

[1] We refer to Paun (2002) for basic information in this area, to Păun et al. (2010) for a comprehensive presentation and the web site http://ppage.psystems.eu for the up-to-date information.

computing introduced by Paun (2000).[2] This model of computation is inspired by the structure and functioning of cells as living organisms able to process and generate information. In particular, it focuses on membranes, which are involved in many reactions taking place inside various compartments of a cell.

The basic idea is inspired by the flow of metabolites between cells of a living tissue or between the organelles in an eucaryotic cell. This flow of metabolites takes place in parallel in nature and it can be interpreted as a flow of information for computational purposes. Instead of a set of few instructions with complex data structures, the computation steps in a membrane computing device are regulated by a set of rules with a notation close to biochemical reactions.

Membrane computing devices are called P systems. They are distributed and have a high degree of parallelism at two levels. On the one hand, several objects can evolve simultaneously in the same membrane and, on the other hand, different membranes can evolve in parallel. Such degree of autonomy and the possibility of locally encapsulating the local information needed for the next step of computation make these devices suitable for modelling the geometry of modular self-reconfigurable robots.

The second bio-inspired tool used in this paper, cellular automata, has been widely used in the literature for the control of self-reconfigurable robots. CA were introduced by John von Neumann (Kari 2005; Wolfram 1994) with the biological motivation of obtaining self-replicating artificial systems. CA are decentralized discrete computational systems which consist of large numbers of simple identical components (cells) placed on an N-dimensional grid with local connectivity defining the *neighbourhood* of a cell. The cells are in one of a finite set of *states*. A discrete global clock is assumed and cells change their states synchronously depending on their own state and the states of the neighbours, as determined by a local update rule. Technically, a CA consists of two components. The first one is a *cellular space*: a lattice of N identical finite-state machines (cells) each with an identical pattern of local connections to other cells, with boundary conditions if the lattice is finite. The second component is a set of transition rules that gives the update state of each cell. These features make CA suitable for dealing with the control of self-reconfigurable robots.[3]

In this paper, CA are used to handle the distributed and parallel motion of decentralized modules. Since the distributed nature of cellular automata and the major properties of the desired controller for modular self-reconfigurable (MSR, for short) robots are complementary, it is possible to develop algorithms based on CA. Cellular automata were introduced to MSR robots by Butler et al. (2001a). Research on distributed control in MSR robots using CA has been productive (Butler et al. 2001a, 2004; Butler and Rus 2003). Bojinov et al. (2000, 2002) combined gradients and CA rules to generate global configurations where rules are based on interactions with surrounding obstacles.

The CA rules in this article are effective in different self-reconfigurations. This efficiency avoids the manual design of CA rules for different tasks (Ostergaard and Lund 2004). In the literature, one can find different efforts for generating an automated design of CA rules, including reinforcement learning-based approaches (Varshavskaya et al. 2008) and evolutionary algorithms (Ostergaard and Lund 2004), but both methods are only effective for relatively small tasks.

The paper is organized as follows: firstly, we recall some basics on membrane computing and show how the cell-like structure of a P system can be interpreted as a 3D configuration of a self-reconfigurable robot. In Sect. 3, we show how the configuration of a tree-like structure can be geometrically represented by a self-reconfigurable robot. Such representation is performed by a cluster-flow locomotion of spare modules inspired on the well-known turtle graphics methods. Finally, several simulations and world-real experiments are shown. The paper finishes with some conclusions and open research lines.

2 Membrane computing

From the birth of membrane computing at the end of the past century, many different computational models have been added to the general framework. If we classify the models according to their topology, there are three basic sets of models, although other approaches are possible (Păun et al. 2010): cell-like P systems, where membranes have a tree-like structure following the inspiration of biological membranes inside an eucaryotic cell; tissue-like P systems, where membranes are placed in the nodes of a general graph as cells in a tissue; and spiking neural P systems, which are inspired by the structure of living neurons in a brain. In this paper we will explore the possibilities of cell-like P systems in order to represent the spatial geometry of modular self-reconfigurable robots.

The basic cell-like P system model consists of a hierarchical structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes divide the Euclidean space into regions, that contain multisets of

[2] See also Păun (1998).

[3] Since cellular automata has been widely used in the literature of self-reconfigurable robots, we assume the reader familiar with these concepts and we stress the use of membrane computing techniques as the main novelty of this paper.

objects (represented by symbols of an alphabet) and/or other membranes. Membranes which do not contain other membranes are called elementary. The intuition behind this membrane structure is taken from biology. A membrane can be seen as a three-dimensional vesicle which is a separator of the region *inside* and the region *outside*. Biological metabolites inside the regions are modelled by object-symbols. Each region, which is defined by a membrane, can contain other symbols or other membranes, so that a P system has exactly one outer membrane, called the skin membrane, and a hierarchical relationship governing all its membranes under the skin membrane. The *information* encapsulated inside each region is encoded in the type of symbols, but also in its multiplicity.

In this paper, the structure of cell-like P systems is used to construct branching structures of self-reconfigurable robots. The rooted tree nature of membranes is a perfect frame to encode the branching structure. Each membrane describes a segment composed of linearly connected modules. The containing membrane structure provides connecting relationship between robotic segments. Relationship between two contained membranes gives the topological orientation of two connected segments.

3 Graphical representations of tree-like structures

In this paper we use the formal framework of membrane computing in order to describe the geometry and the topology of self-reconfigurable robots whose modules can be represented with a tree-like structure. The key points of the representation are the following:

1. Firstly, the geometrical structure of each segment (concerning to length, thickness, color or whatever other features) is represented by a multiset of objects placed in the corresponding membrane.
2. Secondly, the topological relations among the segments are represented by the tree-like membrane structure of the P system. If two segments are joint in the robot, the corresponding membranes are joint in the tree-like structure of the P system, i.e., one of them is contained in the other one.
3. Thirdly, the relative position of a module with respect to its father in the segment will be also encoded with a multiset of objects placed inside the corresponding membrane.

The encapsulation of the information intrinsic to P systems makes possible a natural translation of the idea of module from a physical real robot to the formal computational model. One of the main advantages of this formalism is that no global position is needed in order to describe the topology or geometry of the robot.

3.1 A first example

As an initial example, we can take the stick-man (see Fig. 1 left). It is composed of 12 segments in a tree-like structure. In Fig. 1, a label is associated to each segment (center) and a kind of dual representation of this tree-like structure is depicted (right). In this new representation, segments of the robot are represented by nodes in the graph and there is an edge between the nodes x and y if and only if the segments x and y are joint in the robot. This dual representation will be used for representing the topology of the robot as a cell-like P system structure.

In this way, the tree-like structure of a modular robot has an immediate correspondence with a cell-like membrane structure of a P system. Figure 2 shows a cell-like P system structure associated to the stick man in Fig. 1 (left). Let us remark that this cell-like P system structure takes the vertex a as the root of the tree and hence, the corresponding membrane in the P system structure is the *skin* of the P system, but any other terminal node could be taken as root of the tree.

Obviously, we can obtain the tree-like graph in Fig. 1 (right) from the P system membrane structure in Fig. 2, but if we want to obtain the original stick man in Fig. 1 (left), we must place symbols in the membrane structure which encode the geometric features. Such combination of membrane structure plus the symbols associated to each membrane is called a *configuration* in membrane computing. In this example, we choose the symbol F for representing a length unit. For the sake of simplicity, in this example the unique feature of the segment of the robot described by symbols is the length. Nonetheless, many other features as the width or the color can also been described by multisets of symbols.

According to the membrane computing theory, several copies of a symbol can appear in a membrane. The number of copies of F in a membrane will represent the length of the segment. Instead of a global position of each segment, a representation of the *relative position* of a segment with

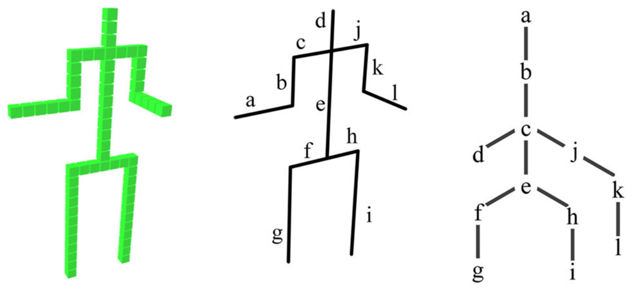


Fig. 1 A stick man configuration, the corresponding scheme graph and an associated labelled tree structure

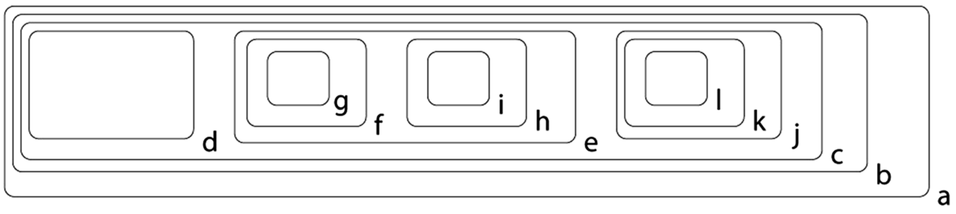


Fig. 2 A P system membrane structure for the stick man

respect to its father in the tree-like representation is proposed. In this way, we borrow ideas from Abelson and DiSessa (1981) and Prusinkiewicz and Lindenmayer (1990).

Let us consider the set of vectors $(\mathbf{H}, \mathbf{L}, \mathbf{U})$, with unit length, perpendicular to each other and satisfying $\mathbf{H} \times \mathbf{L} = \mathbf{U}$. A new vector $(\mathbf{H}', \mathbf{L}', \mathbf{U}')$ can be obtained from the vector $(\mathbf{H}, \mathbf{L}, \mathbf{U})$ by using a rotation matrix $\mathbf{R}$

$$(\mathbf{H}', \mathbf{L}', \mathbf{U}') = (\mathbf{H}, \mathbf{L}, \mathbf{U})\mathbf{R}$$

Rotations by angle α about vectors $\mathbf{U}$, $\mathbf{L}$ and $\mathbf{H}$ are represented by

$$\mathbf{R}_U(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$\mathbf{R}_L(\alpha) = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix}$$
$$\mathbf{R}_H(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

Figure 3 illustrates the rotation around the axis. In this paper, we only consider the angle $\alpha = \pi/2$, but the construction can be made in general. By fixing $\alpha = \pi/2$, we have the matrices

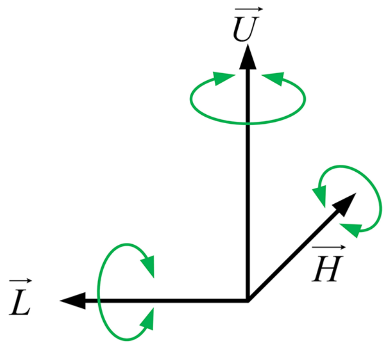


Fig. 3 Rotation around axis

Since all the rotations in this example are $\pi/2$ rotations, it suffices to introduce the symbols $\mathbf{R}_L$, $\mathbf{R}_U$ and $\mathbf{R}_H$ into the membranes.[4] The occurrence of such symbol in a membrane will be interpreted as the rotation angle of the corresponding robot segment with respect its segment father in the like-tree structure. In a similar way that with the F symbol, the multiplicity has also a associated meaning. In a natural way, we will consider that the rotation is applied as many times as the number of copies. In such way, we can represent rotations of $\pi/2$, π or $-\pi/2$ by considering one, two or three copies of the symbol.

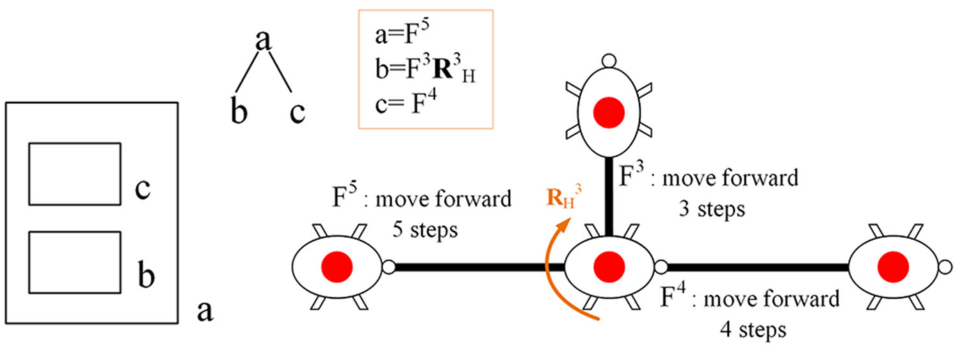
In this way, the general position of the stick man, can be encoded in a P system configuration by considering the membrane structure shown in Fig. 2 and adding to the membrane i in the membrane structure the multiset of symbols w_i (the superscripts denote the multiplicity):

$$\begin{array}{llll} w_a = F^6 & w_b = F^4\,\mathbf{R}_H^3 & w_c = F^3\,\mathbf{R}_H & w_d = F^3\,\mathbf{R}_H^3 \\ w_e = F^{10}\,\mathbf{R}_H & w_f = F^3\,\mathbf{R}_H & w_g = F^{11}\,\mathbf{R}_H^3 & w_h = F^3\,\mathbf{R}_H^3\,. \\ w_i = F^{11}\,\mathbf{R}_H & w_j = F^3 & w_k = F^4\,\mathbf{R}_H & w_l = F^5\,\mathbf{R}_L \end{array}$$

$$\mathbf{R}_U = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}_L = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$
$$\mathbf{R}_H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

[4] Let us notice that the *interpretation* of the symbols $\mathbf{R}_L$, $\mathbf{R}_U$ and $\mathbf{R}_H$ as $\pi/2$ rotations is a *choice* suitable for this example. The use of P systems for the abstract representation of the robot is versatile enough for adapting to different topologies and robot structures not presented in this paper. Such versatility allows to identify the symbols $\mathbf{R}_L$, $\mathbf{R}_U$ and $\mathbf{R}_H$ with the rotation angle chosen by the designer or by taking new symbols associate to them different features of the robot.

Fig. 4 Turtle interpretation of a simple P system


a
b c
c
b
a
a=F5
b=F3R3H
c= F4
F5 : move forward 5 steps
RH3
F3 : move forward 3 steps
F4 : move forward 4 steps

4 Geometrical interpretation of a P system configuration

The structure of membranes in a cell-like P system is a tree-like graph. Such graph does not have an intrinsic geometric interpretation, but we can add such interpretation by giving a geometric meaning to the objects placed inside the membranes. This has been the starting point of view of several approaches which give a geometrical interpretation to the symbols in a P system in order to obtain a two dimensional representation of higher plants (Georgiou et al. 2006; Romero-Jiménez et al. 2006; Rivero-Gil et al. 2011), where some basic properties, such as thickness and length of the branches, are represented by symbol placed in the membranes.

Given a P system configuration, the membrane structure and the multisets placed in the membranes encoded all the needed information for settling the features of the models in the robot and their relative position. Nonetheless, from a methodological point of view, such information must be interpreted in order to have a 3D model of the robot. A simple model to graphically represent a membrane structure is to make a depth-first search of it, drawing, for each membrane containing the object F, a segment of length $m \times l$, where m is the multiplicity of F and l is a length unit. This segment is drawn rotated with respect to the segment corresponding to the parent membrane with an angle of $n \times \delta$, where n is the multiplicity of objects $\mathbf{R}_i$, $i \in \{H, L, U\}$.

Obtaining a 3D model from a P system configuration can be made by using different methods. In this paper, the well-known turtle interpretation (Prusinkiewicz 1986; Zhu et al. 2017) is considered. It was originally used for graphical interpretation of formal languages. The basic idea of this interpretation is quite intuitive. We only recall here some basics principles.

A turtle is placed on an N-dimensional space (usually, $N \in \{2, 3\}$) facing in a certain direction. The turtle can move and its movements are determined by a simple object language. The interpretation of the symbols can be extremely complex, but in its basic version they only control the number of straightforward steps and the angle and direction of turns. In this way, turtle interpretation is an appropriate tool for obtaining a 3D model of a P system configuration.

As an illustrative example, let us consider the P system with three membranes shown in Fig. 4. It has three membranes, the skin (labeled by a) and two elementary membranes (labelled by b and c). The multisets placed in the membranes are $w_a = F^5$, $w_b = F^3 R_U^3$ and $w_c = F^4$, respectively. Figure shows the turtle interpretation of the P system configuration, with three segments of length 5, 3 and 4 determined by the multiplicity of the symbol F. The symbol $\mathbf{R}_U$ determines the turn angle of the second segment with respect to the segment represented by its upper membrane in the cell-like structure.

5 Cluster-flow locomotion of spare modules

In this paper, the final configuration is obtained by a cluster-flow locomotion of the modules of the robot.[5] These modules move from the initial configuration to the final one determined by a P system configuration. A segment of the robot determined by a membrane containing n objects F will be built by n modules in a row. Modules have three kinds of states during the interpretation:

- *Turtle module* Modules that do the moving search work as the turtle.
- *Spare module* Modules that can move to other areas to continue the growth of robotic structures.
- *Finalized module* Modules that have reached the final position will not move any more.

Turtle modules do the turtle search work according to inner objects in membrane configuration, as shown in Fig. 5. Connected modules at the moving direction receive a string of objects in membrane configuration from former turtle modules and become new turtle modules. New turtle

[5] A detailed description of the cluster-flow locomotion is out of the scope of this paper. A good introduction can be found, e.g., in Fitch and Butler (2007) or Butler et al. (2002).

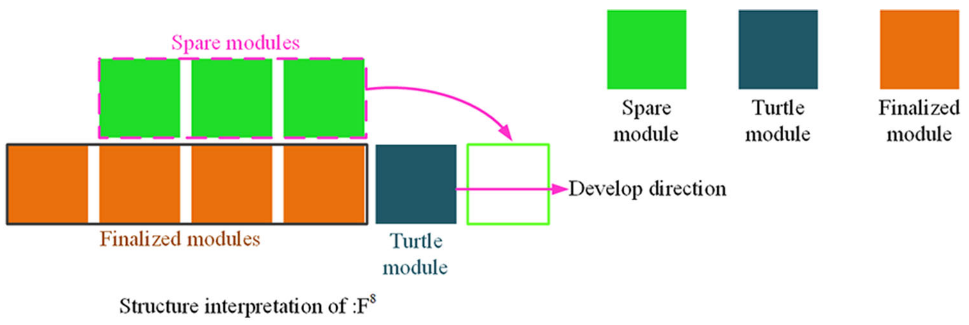


Fig. 5 Robotic segments develop by constantly attaching new modules at neighboring lattice position in develop direction

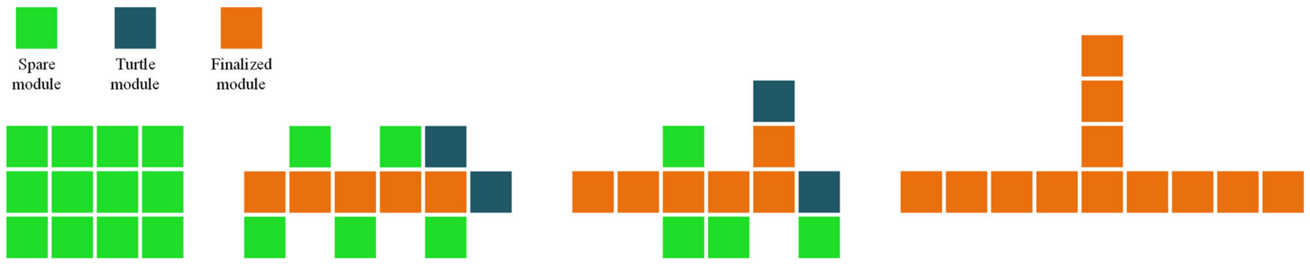


Fig. 6 Turtle interpretation in 3D lattice surroundings

modules receive P system objects by reducing one F. When all neighbouring lattices meet the membrane configuration description, the turtle module changes to finalized module as a fixed part of the reconfiguration result. This locomotion allows the robot to reach the final configuration for totally connected robots. Segments correspond to turtle traces in 3D interpretation.

A distributed localization strategy is used in this decentralized control mechanism. In decentralized control, there is no global map available for local modules to search for their global position and orientation in the whole robotic system. Local modules can only take the relative orientation and position to connected neighbors. During the turtle interpretation in robotic system, a turtle module move by continually attaching spare modules to the neighboring lattice at the moving direction. For their global state, new attached modules take the relative position and orientation to the connected father module, which is the former turtle module. Directly connected modules can determine relative orientation through local communication on connecting surface.

Spare modules can move on the surface of other modules, including finalized modules. Robotic segments develop in the growing style by constantly attaching new modules to the developing front. Figure 6 shows the evolution of the modules from an initial configuration to a final configuration determined by the simple P system shown in Fig. 4. Video attachments (http://www.cs.us.es/～naranjo/MC_SRR/) record the corresponding self-reconfiguration process.

CA is used for the cluster-flow locomotion of spare modules as shown in Zhu et al. (2015). In order to get a computational model for controlling the movement of the modules, a set of CA rules is designed, which only contains two rules. This set of rules is obviously simpler in numbers than the set presented in Butler et al. (2002). Figure 7 shows a scheme of both rules. The scheme on the left will be used for representing a turn of 90 degrees of one of the modules. The scheme on the right represents a one module length movement of a module along the moving direction. The efficiency of designed rules is verified in latter simulations through successful self-reconfiguration motion.

The gradient attraction strategy (Zhu et al. 2015; Stoy 2006) is used to provide moving directions for decentralized modules. Gradient information transferred between connected modules through local communication (Naz et al. 2018) on the connecting surface. Each connecting surface receives gradient value from directly connected neighboring modules. The surface with no connecting module receives 0 as the gradient value. Modules take the

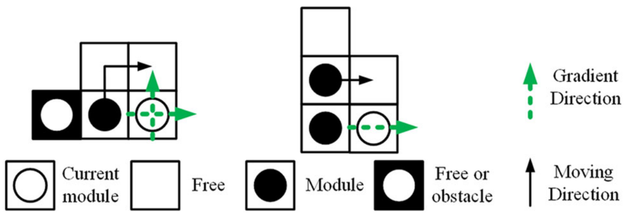


Fig. 7 A simplified set of CA rules for SCM

average value from all 4 (2D lattice modules) or 6 (3D lattice modules) connecting surfaces as the gradient value to fix the moving direction. And modules also transferred this average value to neighboring modules. At the same time, During the movement of moving modules, all their neighboring modules must keep at least one connecting path to the current module. This strategy (Zhu et al. 2015) is used for maintaining the global connection during the locomotion (when several modules can move independently and simultaneously).

6 Simulation and experimental results

The proposed control mechanism is absolutely distributed and it combines CA and membrane computing. In this section we show how these ideas can be implemented in two different ways. Firstly, we present software simulations for 3D modular robots. Secondly, some 2D experiments performed in a hardware lab are presented.

6.1 Simulations

The decentralized nature of modular robots requires the simulator to be absolutely distributed. Our simulation environment is built using Microsoft Robotics Developer Studio[6] and its Decentralized Software Services (DSS), which match with the distributed nature of modular robots. Each module, a sliding cube model (SCM) (Fitch et al. 2003), is handled as an independent thread in the simulator. The control of the motion of each module is independent of the others and several modules can move independently.

Turtle modules attracting new modules to the moving direction through gradient attraction. The interpretation of membrane configuration is a depth-first search process. Several turtle modules can be alive simultaneously in the robot system and send out gradient information. Gradient information spreads out in the robot system through local communication. In this work, we use the decreasing gradient with turtle modules as gradient source. So spare modules move climbing by gradient attraction with local motion controller by CA.

As an illustrative example, Fig. 8 shows the software simulation of the self-reconfiguration process of the stick-man shown in Fig. 1, by using the P system configuration as target configuration whose membrane structure is shown in Fig. 2.

The proposed method is convergent by using membrane computing construction and turtle interpretation. Because the interpretation process is serial by attaching modules one by one. The convergence state is recorded by the

[6] https://msdn.microsoft.com.

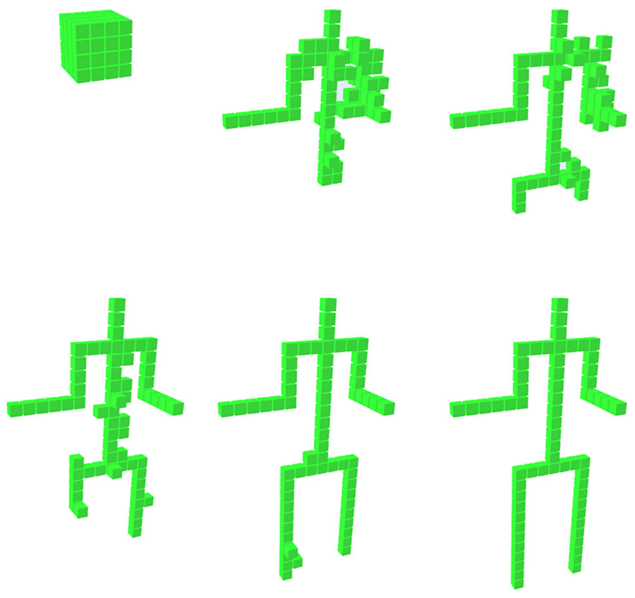


Fig. 8 Reconfiguration process of the stick man

completion of topological description. Using the membrane structure in Fig. 2, two simulations are done in robotic systems with 193 and 321 modules, as shown in Figs. 9 and 10. The stick-man configuration in Fig. 1 are repeated as module numbers increasing, just like the cell division. Each simulation is repeated 50 times and all simulations are convergent to desired configuration.

The proposed method is scalable by using both decentralized methods. Locomotion of spare modules is scalable as all modules moving independently by CA rules. The use of membrane computing is scalable by generating new structures as natural cell division. Multi simulations are done in robotic systems with module numbers increasing. The time steps for convergence of self-reconfiguration process are recorded in each simulation. As shown in Fig. 11, the time step number increases linearly with the number of modules. Video attachments (http://www.cs.us.es/∼naranjo/MC_SRR/) record the corresponding self-reconfiguration process.

6.2 Experiments

In order to perform experimental simulations of the proposed control method, a pivoting cube model robot has been designed. Each module is a 80 mm cube and it holds eight diametrically polarized magnets that are free to rotate along the axial direction, as shown in Fig. 12. Magnets in the four legs form hinge bonds for relative pivoting between modules, as shown by the orange arrows in the figure. Magnets on four face edges help aligning neighboring modules as shown by the black arrows. Modules can communicate with connected modules through onboard

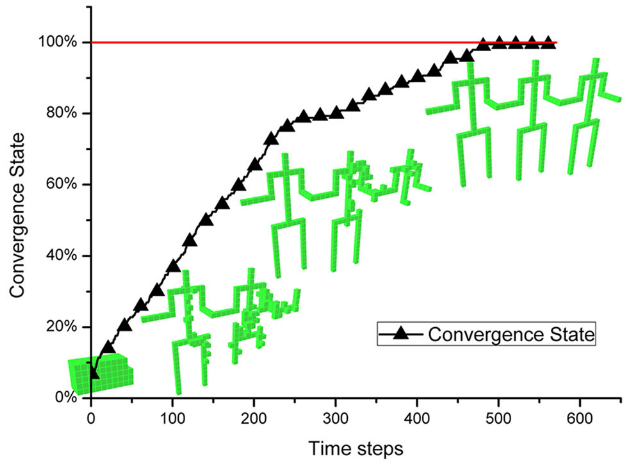


Fig. 9 Convergence of the self-configuration with 193 modules

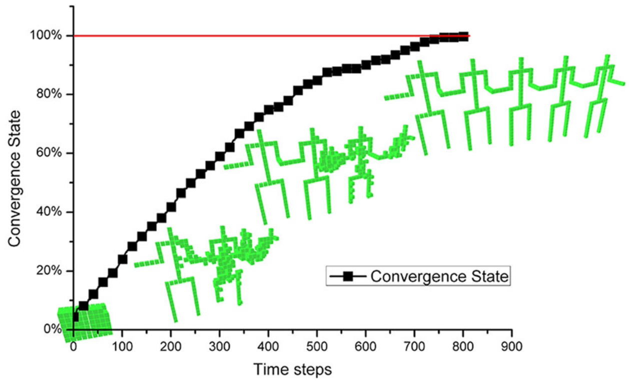


Fig. 10 Convergence of the self-reconfiguration with 321 modules

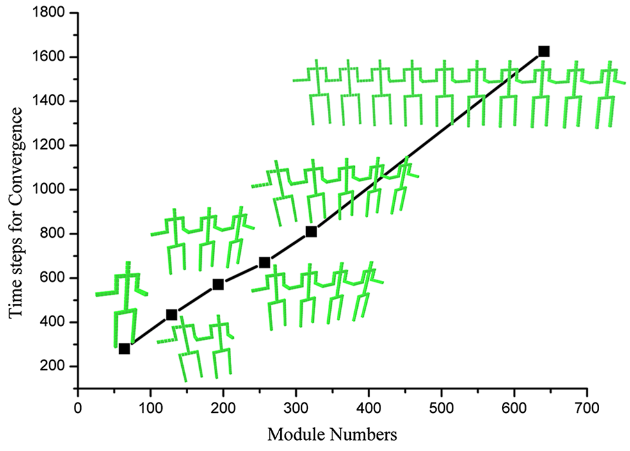


Fig. 11 Scalability of the proposed method for self-reconfiguration of modular robots

infrared sensor on each connecting face. For more details about this robotic module can refer to the site web (https://sites.google.com/site/modularrobots/).

Each module is able to perform 2D movements using inertial servo and DC motor, as shown in Fig. 13. A servo

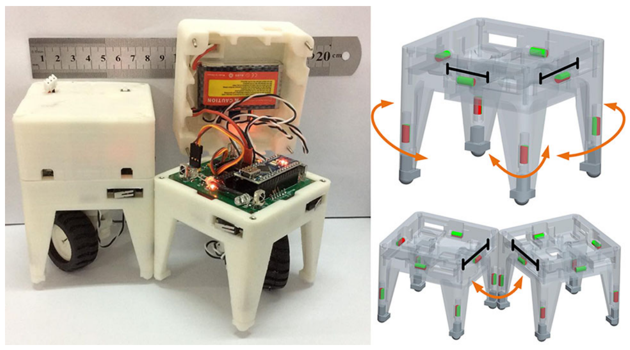


Fig. 12 A 2D pivoting cube model

located inside each module, is used to adjust the orientation of DC motor pointing to one of the four edges. The DC motor connected with a wheel, determines the pivoting direction of modules centring on the servo selected edges. This module in 2D (Piranda et al. 2013) realizes the motion ability of simulating sliding cube model (SCM) and the M-Block (Romanishin et al. 2013). A group of modules can reconfigure autonomously through this rolling locomotion and move as a whole robot using the wheels.

The reconfiguring framework designed in this paper has been verified on a MSR robot. Next, two hardware experiments are presented. In order to verify the convergence of proposed method, each experiment is repeated ten times. Robots in all experiments converge to the desired configuration by given P system.

Figure 14 shows a first hardware experiment. The robot consists of eight modules. The initial configuration is a 4×2 rectangle and the final configuration is provided by a cell-like P system configuration. Figure 14 (left) shows the membrane structure composed by the skin and two elementary membranes. Such simple membrane structure can be interpreted as a tree with the label a for the root vertex and b and c as leaves vertices. Such P system configuration contains all the information of the final desired configuration of the robot. The top (right) scheme in the figure shows a turtle interpretation of such configuration and finally, the bottom (right) six pictures shows the evolution of the robot from the initial configuration to the final one. Video attachments (http://www.cs.us.es/~naranjo/MC_SRR/) record the corresponding self-reconfiguration process.

Figure 15 shows another simple P system configuration with three elementary membranes placed in the region surrounded by the skin, together with the tree structure and the multisets placed in the membranes. Starting from a 3×3 configuration of 9 modules, Fig. 16 shows the evolution from the initial configuration to the one determined by the P system. Video attachments (http://www.cs.us.es/~naranjo/MC_SRR/) record the corresponding self-reconfiguration process.

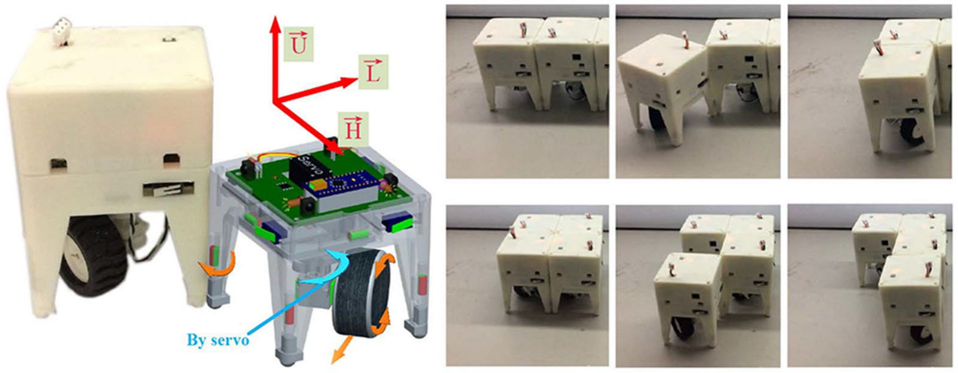
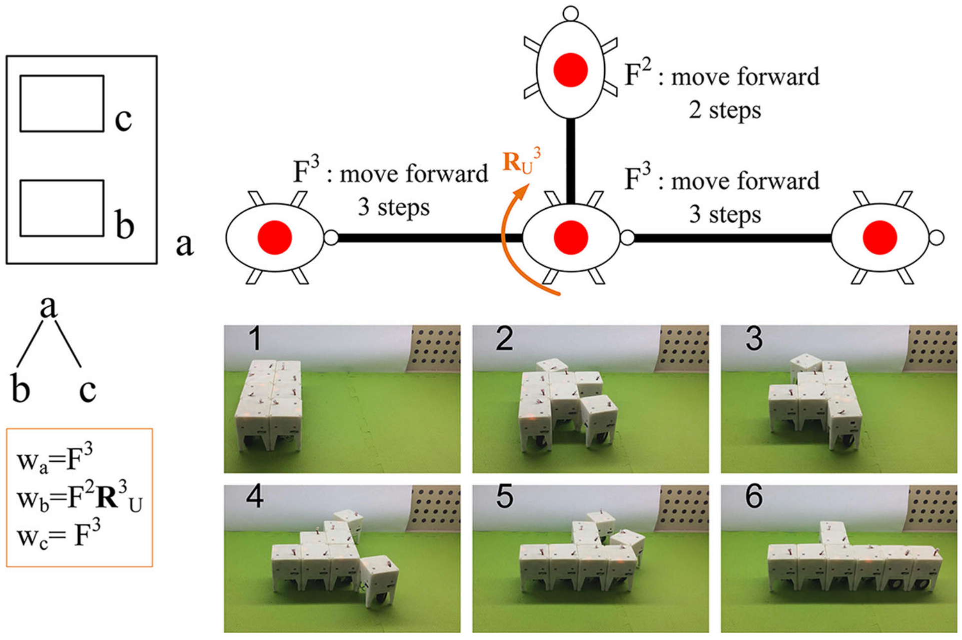


Fig. 13 The basic pivoting motion of modules



c
b
a
a
b c
$w_a=F^3$
$w_b=F^2\mathbf{R}^3_U$
$w_c= F^3$
F^2 : move forward 2 steps
F^3 : move forward 3 steps
R_U^3
F^3 : move forward 3 steps
1
2
3
4
5
6

Fig. 14 Self-reconfiguration of an 8-module robot with a simple P-system

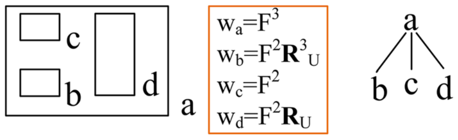


Fig. 15 P system configuration for a cross shape

7 Conclusions

The control of self-reconfigurable robots in an extremely hard task in the interplay of many different research disciplines. Such robots are distributed devices where each module has only local information and there is no central unit to process the information provided by the modules. Each module processes this information locally and independently from the other modules. Several modules can move simultaneously in order to reach the final configuration. Robotic configuration is not determined by a global position in a 3D space, but is only determined by the relative position among modules. To achieve this goal, many different ideas coming from different research areas are brought together. Among them, the partition of the

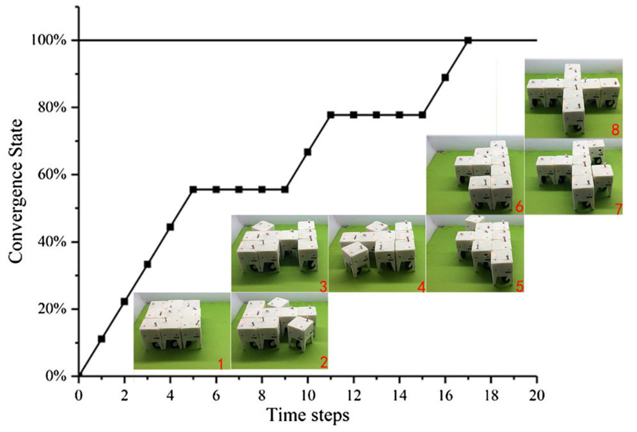


Fig. 16 Experiment with 9 modules from lattice structure to a cross shape

Euclidean space into similar tiles where a module can be *alive* or not; a turtle interpretation for giving a *dynamic* meaning to a set of *static* symbols representing a configuration or a gradient attraction strategy for moving independent modules in order to achieve a final configuration.

In this paper, we provide a new contribution to this big target by considering how ideas from membrane computing can shed a new light to the local representation of the information. The framework presented in this paper can be the starting point for further research from both theoretical and practical sides, enriching each other with ideas coming from the self-configurable robots and from membrane computing. Such open research lines involve new developments in the application of the well-stablished theoretical framework of membrane computing for the abstract representation of robots (and hence a deeper understanding of the theoretical possibilities). and also, from the practical side, the development of new abilities of physical robots inspired by the local relative position of the modules and the local encapsulation of the information. Many problems remain open, for example the automatic design of P-system for a given target robotic configuration. The study of how problems and techniques from both research areas can provide new solutions on the other side is matter of future research.

Acknowledgements The work reported in this paper is supported by the Joint Research Fund (U1713201) between the National Natural Science Foundation of China (NSFC) and Shen Zhen. And the work is also supported by the State Key Laboratory of Robotics (2018-O10).

References

Abelson H, DiSessa AA (1981) Turtle geometry: the computer as a medium for exploring mathematics. The MIT Press series in artificial intelligence. MIT Press, Cambridge **(edition broche 1986)**

Baca J, Woosley B, Dasgupta P, Nelson CA (2017) Configuration discovery of modular self-reconfigurable robots: real-time, distributed, IR+XBee communication method. Robot Auton Syst 91:284–298

Bojinov H, Casal A, Hogg T (2000) Emergent structures in modular self-reconfigurable robots. In: Proceedings of the 2000 IEEE international conference on robotics and automation, ICRA 2000, April 24–28, 2000, San Francisco, CA, USA. IEEE, pp 1734–1741. https://doi.org/10.1109/ROBOT.2000.844846

Bojinov H, Casal A, Hogg T (2002) Multiagent control of self-reconfigurable robots. Artif Intell 142(2):99–120. https://doi.org/10.1016/S0004-3702(02)00272-2

Butler ZJ, Rus D (2003) Distributed locomotion algorithms for self-reconfigurable robots operating on rough terrain. In: Proceedings of the IEEE international symposium on computational intelligence in robotics and automation: computational intelligence in robotics and automation for the New Millennium, CIRA 2003, Kobe, Japan, July 16–20, 2003. IEEE, pp 880–885. https://doi.org/10.1109/CIRA.2003.1222296

Butler Z, Kotay K, Rus D, Tomita K (2001a) Cellular automata for decentralized control of self-reconfigurable robots. In: Proceedings of the IEEE ICRA workshop on modular robots, pp 21–26

Butler Z, Kotay K, Rus D, Tomita K (2001b) Cellular automata for decentralized control of self-reconfigurable robots. In: Proceedings of the IEEE ICRA workshop on modular robots, pp 21–26

Butler Z, Kotay K, Rus D, Tomita K (2002) Generic decentralized control for a class of self-reconfigurable robots. In: IEEE international conference on robotics and automation, 2002. Proceedings. ICRA '02, vol 1, pp 809–816. https://doi.org/10.1109/ROBOT.2002.1013457

Butler ZJ, Kotay K, Rus D, Tomita K (2004) Generic decentralized control for lattice-based self-reconfigurable robots. Int J Robot Res 23(9):919–937. https://doi.org/10.1177/0278364904044409

Christensen DJ (2006) Evolution of shape-changing and self-repairing control for the ATRON self-reconfigurable robot. In: Proceedings of the 2006 IEEE international conference on robotics and automation, ICRA 2006, May 15–19, 2006, Orlando, Florida, USA. IEEE, pp 2539–2545. https://doi.org/10.1109/ROBOT.2006.1642084

Fitch R, Butler ZJ (2007) Scalable locomotion for large self-reconfiguring robots. In: 2007 IEEE international conference on robotics and automation, ICRA 2007, 10–14 April 2007, Roma, Italy. IEEE, pp 2248–2253. https://doi.org/10.1109/ROBOT.2007.363654

Fitch R, McAllister R (2010) Hierarchical planning for self-reconfiguring robots using module kinematics. In: Martinoli A, Mondada F, Correll N, Mermoud G, Egerstedt M, Hsieh MA, Parker LE, Støy K (eds) Distributed autonomous robotic systems—the 10th international symposium, DARS 2010, Lausanne, Switzerland, November 1–3, 2010, vol 83 of Springer Tracts in Advanced Robotics. Springer, Berlin, pp 477–490. https://doi.org/10.1007/978-3-642-32723-0_34

Fitch R, Butler Z, Rus D (2003) Reconfiguration planning for heterogeneous self-reconfiguring robots. In: International conference on intelligent robots and systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ, vol 3, IEEE. pp 2460–2467. https://doi.org/10.1109/IROS.2003.1249239

Fitch R, Butler ZJ, Rus D (2005) Reconfiguration planning among obstacles for heterogeneous self-reconfiguring robots. In: Proceedings of the 2005 IEEE international conference on robotics and automation, ICRA 2005, April 18–22, 2005, Barcelona, Spain. IEEE, pp 117–124. https://doi.org/10.1109/ROBOT.2005.1570106

Fukuda T, Nakagawa S (1988) Approach to the dynamically reconfigurable robotic system. J Intell Robot Syst 1(1):55–72. https://doi.org/10.1007/BF00437320

Funiak S, Pillai P, Ashley-Rollman MP, Campbell J, Goldstein SC (2009) Distributed localization of modular robot ensembles. Int J Robot Res 28(8):946–961. https://doi.org/10.1177/0278364909339077

Georgiou A, Gheorghe M, Bernardini F (2006) Membrane-based devices used in computer graphics. In: Ciobanu G, Păun G, Pérez-Jiménez MJ (eds) Applications of membrane computing, Natural Computing Series, Springer, Berlin, pp 253–281. https://doi.org/10.1007/3-540-29937-8_9

Gilpin K, Rus D (2012) What's in the bag: a distributed approach to 3D shape duplication with modular robots. In: Roy N (ed) Robotics: science and systems VIII. University of Sydney, Sydney

Goldstein SC, Campbell J, Mowry TC (2005) Programmable matter. IEEE Comput 38(6):99–101. https://doi.org/10.1109/MC.2005.198

Hou F, Shen W (2014) Graph-based optimal reconfiguration planning for self-reconfigurable robots. Robot Autonom Syst 62(7):1047–1059. https://doi.org/10.1016/j.robot.2013.06.014

Jones CV, Mataric MJ (2003) From local to global behavior in intelligent self-assembly from. In: Proceedings of the 2003 IEEE international conference on robotics and automation, ICRA 2003, September 14–19, 2003, Taipei, Taiwan. IEEE, pp 721–726. https://doi.org/10.1109/ROBOT.2003.1241679

Kari J (2005) Theory of cellular automata: a survey. Theor Comput Sci 334(1–3):3–33. https://doi.org/10.1016/j.tcs.2004.11.021

Naz A, Piranda B, Bourgeois J, Goldstein SC (2018) A time synchronization protocol for large-scale distributed embedded systems with low-precision clocks and neighbor-to-neighbor communications. J Netw Comput Appl 105:123–142

Ostergaard E, Lund HH (2004) Controlling self-reconfiguration using cellular automata and gradients. In: Proceedings of the 8th international conference on intelligent autonomous systems (IAS-8), pp 291–298

Pillai P, Campbell J, Kedia G, Moudgal S, Sheth K (2006) A 3D fax machine based on claytronics. In: 2006 IEEE/RSJ international conference on intelligent robots and systems, IROS 2006, October 9–15, 2006, Beijing, China. IEEE, pp 4728–4735. https://doi.org/10.1109/IROS.2006.282264

Piranda B, Laurent GJ, Bourgeois J, Clvy C, Mbes S, Fort-Piat NL (2013) A new concept of planar self-reconfigurable modular robot for conveying microparts. Mechatronics 23(7):906–915. https://doi.org/10.1016/j.mechatronics.2013.08.009 (1.Fractional Order Modeling and Control in Mechatronics 2. Design,control, and software implementation for distributed {MEMS} (dMEMS))

Prusinkiewicz P (1986) Graphical applications of L-systems. In: Proceedings on graphics interface '86/vision interface '86. Canadian Information Processing Society, Toronto, pp 247–253

Prusinkiewicz P, Lindenmayer A (1990) The algorithmic beauty of plants, Virtual Laboratory. Springer, Berlin

Păun G (1998) Computing with membranes, Technical Report 208. Turku Centre for Computer Science, Turku

Păun G (2000) Computing with membranes. J Comput Syst Sci 61(1):108–143. https://doi.org/10.1006/jcss.1999.1693

Păun G (2002) Membrane computing. An introduction. Springer, Berlin

Păun G, Rozenberg G, Salomaa A (eds) (2010) The Oxford handbook of membrane computing. Oxford University Press, Oxford

Rivero-Gil E, Gutiérrez-Naranjo MA, Romero Jiménez Á, Riscos-Núñez A (2011) A software tool for generating graphics by means of P systems. Nat Comput 10(2):879–890. https://doi.org/10.1007/s11047-010-9198-9

Romanishin J, Gilpin K, Rus D (2013) M-blocks: momentum-driven, magnetic modular robots. In: 2013 IEEE/RSJ international conference on intelligent robots and systems, Tokyo, Japan, November 3–7, 2013. IEEE, pp 4288–4295. https://doi.org/10.1109/IROS.2013.6696971

Romero-Jiménez Á, Gutiérrez-Naranjo MA, Pérez-Jiménez MJ (2006) Graphical modeling of higher plants using P systems. In: Hoogeboom HJ, Păun G, Rozenberg G, Salomaa A (eds) Workshop on membrane computing, vol 4361 of lecture notes in computer science. Springer, Berlin, pp 496–506

Rubenstein M, Shen W (2010) Automatic scalable size selection for the shape of a distributed robotic collective, In: 2010 IEEE/RSJ international conference on intelligent robots and systems, October 18–22, 2010, Taipei, Taiwan. IEEE, pp 508–513. https://doi.org/10.1109/IROS.2010.5650906

Stoy K (2004) Controlling self-reconfiguration using cellular automata and gradients. In: Proceedings of the 8th international conference on intelligent autonomous systems (IAS-8), pp 693–702

Stoy K (2006) Using cellular automata and gradients to control self-reconfiguration. Robot Autonom Syst 54(2):135–141. https://doi.org/10.1016/j.robot.2005.09.017 **(intelligent Autonomous Systems 8th Conference on IntelligentAutonomous Systems (IAS-8))**

Stoy K (2015a) Lattice automata for control of self-reconfigurable robots. In: Sirakoulis GC, Adamatzky A (eds), Robots and lattice automata, vol 13 of emergence, complexity and computation. Springer, Berlin, pp 33–45. https://doi.org/10.1007/978-3-319-10924-4_2

Stoy K (2015b) Lattice automata for control of self-reconfigurable robots. In: Sirakoulis GC, Adamatzky A (eds) Robots and lattice automata. Springer, Cham, pp 33–45. https://doi.org/10.1007/978-3-319-10924-4_2

Stoy K, Nagpal R (2007) Self-reconfiguration using directed growth. In: Alami R, Chatila R, Asama H (eds) Distributed autonomous robotic systems, vol 6. Springer, Tokyo, pp 3–12. https://doi.org/10.1007/978-4-431-35873-2_1

Stoy K, Brandt D, Christensen DJ (2010) Self-reconfigurable robots: an introduction. In: Intelligent robotics and autonomous agents. MIT Press, Cambridge

Varshavskaya P, Kaelbling LP, Rus D (2008) Automated design of adaptive controllers for modular robots using reinforcement learning. Int J Robot Res 27(3–4):505–526. https://doi.org/10.1177/0278364907084983

Walter JE, Tsai EM, Amato NM (2005) Algorithms for fast concurrent reconfiguration of hexagonal metamorphic robots. IEEE Trans Robot 21(4):621–631. https://doi.org/10.1109/TRO.2004.842325

Web page http://www.cs.us.es/∼naranjo/MC_SRR/

Web page https://sites.google.com/site/modularrobots/

Wolfram S (1994) Cellular automata and complexity: collected papers. Perseus Books Group, Addison-Wesley Pub. Co., c1994

Wu Q, Wang Y, Cao G, Fei Y (2005) Locomotion control of distributed self-reconfigurable robot based on cellular automata. In: Huang D-S, Zhang X-P, Huang G-B (eds), Advances in intelligent computing, vol 3645 of lecture notes in computer science. Springer, Berlin, pp 179–188. https://doi.org/10.1007/11538356_19

Yim M, Shen W-M, Salemi B, Rus D, Moll M, Lipson H, Klavins E, Chirikjian GS (2007) Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robot Autom Mag 14(1):43–52. https://doi.org/10.1109/mra.2007.339623

Yoshida E, Murata S, Kurokawa H, Tomita K, Kokaji S (1998) A distributed reconfiguration method for 3D homogeneous structure. In: Proceedings 1998 IEEE/RSJ international conference on intelligent robots and systems. Innovations in theory, practice and applications, October 13–17, 1998, Victoria, BC, Canada. IEEE, pp 852–859. https://doi.org/10.1109/IROS.1998.727306

Zavlanos MM, Pappas GJ (2008) Dynamic assignment in distributed motion planning with local coordination. IEEE Trans Robot 24(1):232–242. https://doi.org/10.1109/TRO.2007.913992

Zhao J, Wang X, Jin H, Bie D, Zhu Y (2015) Automatic locomotion generation for a ubot modular robot-towards both high-speed and multiple patterns. Int J Adv Robot Syst 12:32

Zhu Y, Bie D, Iqbal S, Wang X, Gao Y, Zhao J (2015) A simplified approach to realize cellular automata for ubot modular self-reconfigurable robots. J Intell Robot Syst 79(1):37–54. https://doi.org/10.1007/s10846-014-0084-z

Zhu Y, Bie D, Wang X, Zhang Y, Jin H, Zhao J (2017) A distributed and parallel control mechanism for self-reconfiguration of modular robots using L-systems and cellular automata. J Parallel Distrib Comput 102:80–90

Zoppi M (2011) Self-reconfigurable robots: an introduction. Ind Robot Int J. https://doi.org/10.1108/ir.2011.04938daa.015