# Implementing Configuration Dependent Gaits in a Self-Reconfigurable Robot

K. Støy[1][*], W.-M. Shen[2], and P. Will[2]
kaspers@mip.sdu.dk, shen@isi.edu, and will@isi.edu

[1]The Maersk Mc-Kinney Moller Institute for Production Technology
Campusvej 55, DK-5230 Odense M, Denmark

[2]USC Information Sciences Institute and Computer Science Department
4676 Admiralty way, Marina del Rey, CA 90292, USA

## Abstract

*In this paper we examine locomotion in the context of self-reconfigurable robots. Self-reconfigurable robots are robots built from many connected modules. A self-reconfigurable robot can change its shape and configuration by changing the way these modules are connected. The focus of this paper is to understand how several locomotion gaits can be represented in such a robot and how the robot can select one of these gaits depending on its configuration. We implement a control system based on role based control in a physical self-reconfigurable robot built from seven modules. In several experiments we successfully demonstrate that the robot can change from a sidewinder snake gait to a quadruped walking gait when the robot is manually reconfigured from a chain to a quadruped configuration. We conclude that role based control is a promising control method for controlling locomotion of self-reconfigurable robots.*

## 1 Introduction

Self-reconfigurable robots are robots built from potentially many connected modules (see Figure 2 for a photo of such a module or refer to one of several physical realizations of self-reconfigurable robots [7, 8, 10, 11, 12, 13, 14, 15, 16, 22, 25]). Self-reconfigurable robots can autonomously change the way in which these modules are connected and through this self-reconfiguration process change their shape to fit the task-environment. Due to this capability self-reconfigurable robots are useful in task-environments where versatility is of importance [11]. A robot exploring the surface of a planet might for instance self-reconfigure into a snake to explore small caves and later into a rolling track to efficiently cover the distance back to the station.

[*]The work reported here was performed while visiting USC Information Sciences Institute

Another desirable feature of self-reconfigurable robots is robustness [11]. The ideas is that if one module fails another one can replace it. However in order to achieve this the system has to be built from identical modules. It is of course an open question if the system should be built from just one kind of modules or a few, but for the sake of robustness it is important that there are modules to replace the ones which fail. In order to create a versatile and robust self-reconfigurable robot the control system also has to support these features. This implies that modules should run identical programs to make sure they can replace each other. Furthermore distributed control is mandatory to avoid a single point of failure. These constraints on the control system makes the control of self-reconfigurable robots a significant challenge.

Some related work has been focused on how to reconfigure from one given shape to another. Centralized methods exist which plan how to reconfigure from one given shape to another [6, 5, 15] or distributed methods [11, 21]. Also methods exist where the shape emerges due to interaction between the modules and the environment [1]. In this work we focus on locomotion of self-reconfigurable robots. There are two broad classes of locomotion algorithms for self-reconfigurable robots: "water-flow" algorithms where the robot locomotes by having the modules climbing over each other [8, 2, 3]. However, for robots covering large distances these types of locomotion might not be sufficiently fast and energy efficient. Therefore it is important to develop systems which are able to locomote in a fixed configuration. This area is fairly well understood and solutions exist based on gait control tables [23, 24], hormones [16, 17], and roles [18, 19].

The pitfall of studying the control of self-reconfigurable robots in a fixed configuration is that in a fixed configuration it is not important to design a versatile and robust control system in order to make the system work. Therefore there is a risk of producing a control system that does not fit in the context of self-reconfigurable robots. In order to avoid this pitfall we investigate how several locomotion gaits can be represented in one self-reconfigurable system and how the system can change between these gaits during reconfiguration. In particular we extend our previous work by showing how role based control can be used in one system to represent a gait similar to that of a sidewinder snake and a quadruped walker. In role based control each module in the system plays a role. A role controls the motion of the module and takes care of the coordination with connected modules. Which role to play is decided based on both local configuration and the roles being played by neighboring connected modules. Specific gaits are not represented globally, but emerge due to the interaction between role playing modules. This means that as the robot is manually reconfigured the locomotion pattern changes gradually. In experiments with a real self-reconfigurable robot built from seven modules we demonstrate that the robot is able to make a transition from a sidewinder gait to a walker gait when it is manually reconfigured from a chain configuration to a quadruped configuration.

## 2 Role Based Control

Role based control consists of two algorithms. The role playing algorithm controls the motion of a module and the coordination and communication with connected modules. The role selection algorithm selects which role a module plays based on the local configuration and the roles being played by connected modules. We will first describe the role playing algorithm and then the role selection algorithm.

### 2.1 Role Playing

It is assumed that one of the connectors of a module is defined as a parent connector $p \in S$ where S is the set consisting of the connectors of a module. The remaining connectors are defined as child connectors $C = S \backslash \{p\}$. Modules can only be connected by connecting the parent connector to a child connector of another module. Note that the CONRO modules (see Figure 2) cannot physically be connected in any way that breaks these assumptions. The reason being that the female connector can be defined as the parent connector and the three male connectors as child connectors. Furthermore it is assumed that there are no loops in the system. These assumptions essentially limits the allowable configurations to tree configurations with well-defined parent-child relationships.

A role consists of three components. The first component is a function $A(t)$ that specifies the joint angles of a module given an integer $t \in [0 : T]$, where $T$ is the period of the motion and the second component that needs to be specified. The third component is a set of

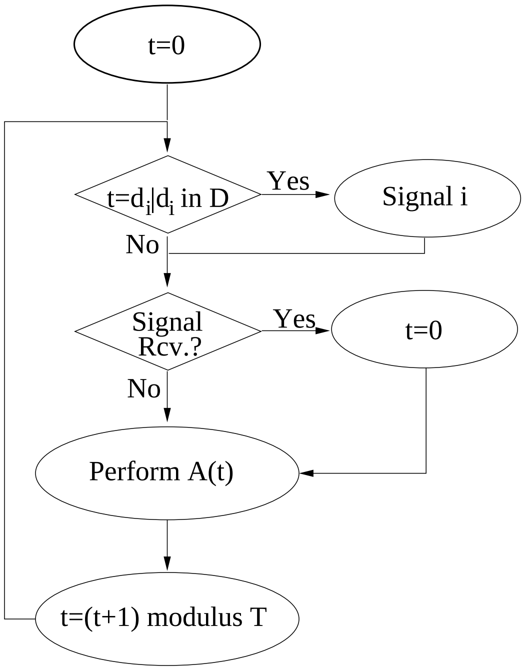

Figure 1: A flow diagram of the role playing algorithm. Refer to Section 2.1 for a description.

delays $D$. A delay $d_i \in D$ specifies the delay between the child connected to connector $i \in C$ and the parent. That is, if the parent is at step $t_{parent}$ the child is at $t_{child} = (t_{parent} - d_i)\ modulus\ T$.

Given these assumptions a module plays a role by following the role playing algorithm outlined in Figure 1. Initially $t$ is set to zero. If it is time to send a signal because $t = d_i$ a signal is sent through connector $i$. If a signal is received from the parent, through connector $p$, $t$ is reset. This insures that the module is delayed as described above compared to the parent. Finally the positions of the servos are updated and $t$ is incremented after which the next iteration is initiated.

In earlier work we have shown that this basic role playing algorithm is sufficient to produce locomotion patterns similar to that of a caterpillar and a sidewinder snake [18].

### 2.2 Role Selection

In more complex locomotion patterns it cannot be assumed that all modules play the same role. This implies that modules need to be able to play different roles and be able to select between them. This introduces the need for a role selection algorithm. The selection is based on the local configuration and which roles connected modules are playing. Before we introduce the role selection algorithm we will introduce some functions. The function $C : S \rightarrow Boolean$ returns true if a module is connected to connector $i \in S$ and false otherwise. The function $NC : S \rightarrow S$ returns the connector of the neighbor to which the connector $i \in S$ of this module is connected. Given that $R$ is the set of roles that a module can play the function $NR : S \rightarrow R$ returns the role being played by the module connected to connector $i \in S$.

The local configuration is specified by: 1) The subset of connectors to which modules are connected $\{i \in S | C(i)\}$ 2) The connector of the parent to which the module is connected $NC(p)$ if $C(p)$. The local configuration is sufficient to select roles in simple configurations.

In configurations where the role cannot only be selected based on the local configuration the role is selected based on the set of roles being played by connected modules $\{r \in R | i \in S \wedge C(i) \wedge R(i) = r\}$. For instance five modules might be connected in a chain to form an arm. In this configuration the modules might have to play the roles of a shoulder, upper arm, elbow, lower arm, and wrist. The three modules in the middle cannot decided which role to play based on the local configuration, because the local configuration is identical for all of them. However if a module finds out that the parent is playing the shoulder role it can select the upper arm role and so on. In a tree configuration the root can determine it is the root based on the local configuration. The root can then instruct its children that they are children of the root and where they are connected. By induction we can see that it is possible to use this mechanism to decide a modules position in the global configuration if needed.

By combing these two selection mechanisms it can be decided as locally as possible what role a module should play which is important if the system is to scale.

## 3 The CONRO Robot

In the experiments reported in Section 5 we use the CONRO self-reconfigurable robot. The CONRO robot has been developed at USC's Information Sciences Institute [4, 9] (see Figure 2). The modules are roughly shaped as rectangular boxes measuring 10cm x 4.5cm x 4.5cm and weigh 100grams. The modules have a female connector located at one end by definition facing south $s$ and three male connectors located at the other end facing east $e$, west $w$, and north $n$ making $S_{conro} = \{s, e, w, n\}$. The parent connector $p$ is defined to be the female connector $s$. Each connector has an infra-red transmitter and receiver used for local communication and sensing. The modules have two controllable degrees of freedom: pitch (up and down) and yaw (side to side). Processing is taken care of by an onboard Basic Stamp 2 processor. The modules have onboard batteries, but these do not supply enough power for the experiments reported here and therefore the modules are powered through cables.

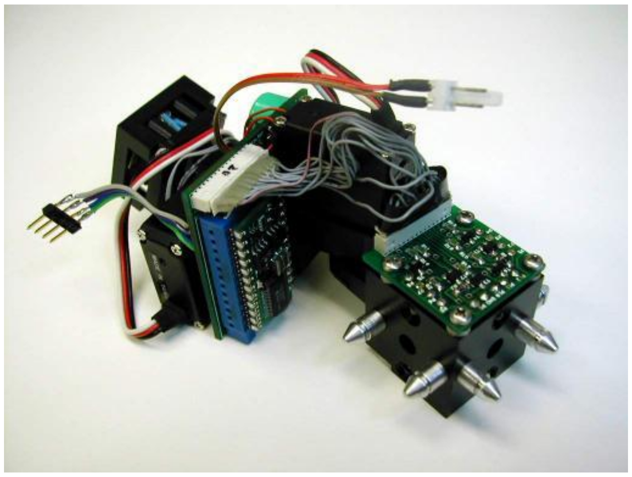

Figure 2: A CONRO module. The male child connectors are at the bottom right corner of the photo. The female connector is hidden from view behind the module in the top left corner.

## 4 Implementation

We will now show how to combine the sidewinder gait and the walking gait previously reported in [18, 19]. In order to perform these gaits four different roles are needed: sidewinder (sw), spine (sp), east leg (eleg), and west leg (wleg). That is $R = \{sw, sp, eleg, wleg\}$. These roles are listed below. The focus of this paper is to understand how these roles are combined and not how to design roles so therefore if you want more details about these role definitions please refer to [18, 19].

The sidewinder role (sw).

$$A(t, sw) = \left\{ \begin{array}{rcl} pitch(t) & = & 20^\circ cos(\frac{2\pi}{T} t) \\ yaw(t) & = & 50^\circ sin(\frac{2\pi}{T} t) \end{array} \right.$$

$$d_{north} = \frac{T}{5}$$

$$T = 180$$

The spine role (sp).

$$A(t, sp) = \left\{ \begin{array}{rcl} pitch(t) & = & 0^\circ \\ yaw(t) & = & 25^\circ \cos(\frac{2\pi}{T} t + \pi) \end{array} \right.$$

$$d_{east} = \frac{T}{4}$$

$$d_{south} = \frac{2T}{4}$$

$$d_{west} = \frac{3T}{4}$$

$$T = 180$$

The east leg role (eleg) is shown below. The west leg role (wleg) is identical except that $t$ is replaced by $2\pi - t$.

$$A(t, eleg) = \begin{cases} pitch(t) & = & 35^\circ \cos(\frac{2\pi}{T}t) - 55^\circ \\ yaw(t) & = & 40^\circ \sin(\frac{2\pi}{T}t) \end{cases}$$
$$T = 180$$

While playing these roles the local configuration and the roles being played by connected modules are monitored.This information is used to decide when to change role. The rules are as described below and shown in Figure 3.

A module playing the sidewinder role changes to the spine role if a module is connected to either the east or the west connector. If it is connected to the east or west side of a module it plays the role of the corresponding leg.

The rules for the spine role are similar to those of the sidewinder role except that if both modules to the sides are detached it changes role to a sidewinder module.

The legs decide to change role to sidewinder or spine if they are connected to the north connector of a module playing one of these roles. Additionally if a leg has a module connected on its north connector it changes role to sidewinder. This last transition is made to make sure that a leg placed at the root of the configuration tree discovers that and plays an appropriate role.

## 5 Experiments

In order to evaluate this system we conducted a series of experiments with a self-reconfigurable robot made from seven CONRO modules. The robot is initially configured into a long chain. The root module is started using an infrared signal. The root module signals its child after $T/5$ and so on. After $6T/5$ corresponding to approximately three seconds all modules are synchronized and take part in the sidewinder gait. The synchronization time is increased by one period per synchronization signal missed. However in the experiments reported this was not observed. It was then recorded how long it takes to move 63cm using the sidewinder gait. The robot is then picked up and by-hand reconfigured into a four legged walker. In each experiment the reconfiguration sequence is different. The reconfiguration sequence for experiment 1 can be seen in Figure 4. When the system is synchronized again the robot is allowed to walk 87cm and this time is also measured. Note that throughout each experiment the modules are not reset. Three snapshots from an experiment is shown in Figure 5.

$$\begin{array}{lll}
\text{sw} & \rightarrow \text{sp}, & \text{if} \quad C(e) \vee C(w) \\
 & \rightarrow \text{wleg}, & \text{if} \quad C(p) \wedge NC(p) = w \\
 & \rightarrow \text{eleg}, & \text{if} \quad C(p) \wedge NC(p) = e \\
\\
\text{sp} & \rightarrow \text{sw}, & \text{if} \quad \neg(C(e) \vee C(w)) \\
 & \rightarrow \text{wleg}, & \text{if} \quad C(p) \wedge NC(p) = w \\
 & \rightarrow \text{eleg}, & \text{if} \quad C(p) \wedge NC(p) = e \\
\\
\text{leg} & \rightarrow \text{sw}, & \text{if} \quad (C(p) \wedge R(p) = sw) \vee C(n) \\
 & \rightarrow \text{sp}, & \text{if} \quad C(p) \wedge R(p) = sp
\end{array}$$

Figure 3: This figure shows the rules used to decide when to change between roles. The roles $R$ are sidewinder ($sw$), spine ($sp$), east leg ($eleg$), and west leg($wleg$). The transitions for the west leg and the east leg are the same and are therefore both represented by $leg$. The CONRO module have the connectors $S = \{e, w, n, s\}$ where the parent connector $p = s$. The function $C : S \rightarrow Boolean$ returns a boolean value that indicates if a modules is connected to the specified connector. The function $NC : S \rightarrow R$ returns the connector of the parent module to which this module is connected. The function $R : S \rightarrow R$ returns the role being played by the module connected to the specified connector.

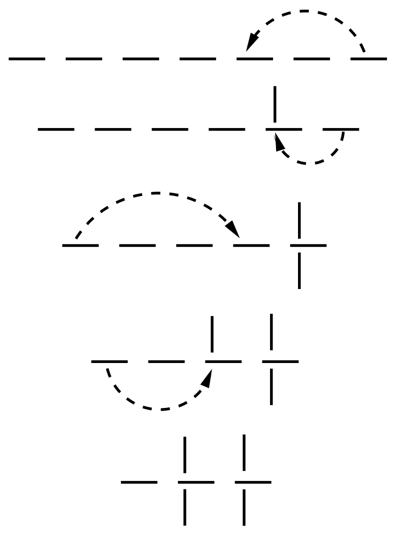

Figure 4: The reconfiguration process of experiment 1. The solid lines represent modules. The robot starts in a chain configuration as shown at the top. The modules are then manually reconfigured as indicated by the dashed arrows until the quadruped configuration shown at the bottom is obtained.

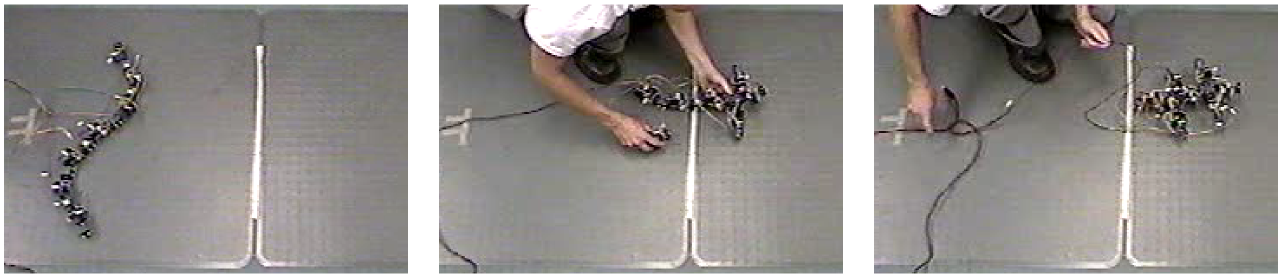

Figure 5: The robot first covers a distance of 63cm configured as a sidewinder (left). The robot is then manually reconfigured into a quadruped walker (middle). Finally the robot walks 87cm (right). Note that the cables only provide power.

We repeated the experiment four times. It took on average 12.3±1.0sec to cover 63cm corresponding to 5.1cm/sec as a sidewinder. It took 36.5±2.4sec to walk 87cm corresponding to 2.4cm/sec. Detailed data can be found in Figure 6. Videos of these experiments can be found on the web page: http://www.isi.edu/conro.

| experiment | sidewinder | walker |
|---:|:---:|:---:|
| 1 | 13 | 25 |
| 2 | 11 | 30 |
| 3 | 13 | 28 |
| 4 | 12 | 30 |
| avg | 12.3 | 36.5 |
| std.dev. | 1.0 | 2.4 |

Figure 6: This figure shows the experimental data for four experiments. The second column indicated the time (seconds) it took for the robot to synchronize and move 63cm using the sidewinder gait. The third column indicates the time (seconds) it to took to locomote 87cm using the walker gait. The last two rows indicate respectively the average and standard deviation of these four experiments.

## 6 Discussion

In the implementation section we saw that it is easy to extend role based control to be able to represent two gaits and a mechanism to select between them. The implementation is minimal which is also indicated by the fact that the main loop is implemented using only 350 lines of Basic Stamp 2 code including code for communication and motor control. The algorithm is also efficient. It manages to keep the modules synchronized only using constant time per period meaning that the locomotion speed is independent of the number of modules as shown in [19]. The time required to synchronize the system after a role change is proportional to $O(h)$ where $h$ is the height of the configuration tree. Note that it is possible and might sometimes might be desirable to have role oscillations occurring.

One concern is the complexity of the role selection conditions. A system could be conceived where the roles of all modules have to change every time the position of one module in the configuration tree is changed. The idea could be to insist on having the optimal gait independent of the shape of the configuration tree. However this complexity is probably not due to the algorithm, but due to the complexity of the problem. The algorithms based on hormones have the same problem [16, 17].

The experiments demonstrate that the system is robust to communication errors and reconfiguration which are important features of algorithms designed for self-reconfigurable robots. It might be worth to try to understand how this extreme robustness comes about. In Shen et al. [17] hormones are created by one module and the actions of the rest of the modules are executed and synchronized as this hormone is propagated through the configuration tree and back. The next hormone is generated when the previous hormone is propagated back to the creator. This means that critical state information is only represented one place in the system - in the hormone. The consequence is that if a hormone is lost the system stops (note that this event is likely to happen during reconfiguration). In role based control this is avoided, because each module essentially acts as the master of its subtree and signals are resent every period. In Yim [23, 24] modules have IDs and pick actions based on this ID which means each module has to be placed in a specific position in the configuration tree in order for the system to work. Furthermore the system is synchronized by a centralized master which means that if the master fails the system fails. This leads us to conclude that role based control gains its robustness from the distributed representation of the state of the gait and the fact that the modules are not assumed to have IDs.

In the work presented here roles are selected based on changes in the local configuration and role changes of connected modules. However roles can also be changed due to sensor input as investigated in [20]. Another option is to use time to trigger a role change.

In the current implementation the robot is reconfigured by hand. This is of course not satisfiable. We are currently working on using role based control to control the self-reconfiguration process as well. This would give a simple uniform algorithm able to control both self-reconfiguration and locomotion.

## 7 Conclusion

In this paper we have investigated how multiple gaits can be implemented in the same system using role based control. We have demonstrated that a self-reconfigurable robot made from seven CONRO modules is able to change from a sidewinder gait to a walker gait when the robot is manually reconfigured from a chain to a quadruped walker. The reconfiguration is done by hand, but without resetting or reprogramming the modules. Finally we have discussed that the key aspects of the system which make it possible to change between gaits include a distributed representation of the state and a system that does not rely on IDs.

## Acknowledgments

This research is supported by the DARPA contract DAAN02-98-C-4032, the AFOSR contract F49620-01-1-0020, the EU contract IST-20001-33060, and the Danish Technical Research Council contract 26-01-0088.

## References

[1] H. Bojinov, A. Casal, and T. Hogg. Emergent structures in modular self-reconfigurable robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, volume 2, pages 1734 –1741, San Francisco, California, USA, 2000.

[2] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Cellular automata for decentralized control of self-reconfigurable robots. In *ICRA 2001 Workshop on Modular Self-Reconfigurable Robots*, Seoul, Korea, 2001.

[3] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for a class of self-reconfigurable robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'02)*, pages 809–815, Washington, D.C., USA, 2002.

[4] A. Castano, R. Chokkalingam, and P. Will. Autonomous and self-sufficient conro modules for reconfigurable robots. In *Proc. of the 5th Int. Symposium on Distributed Autonomous Robotic Systems*, pages 155–164, Knoxville, Texas, USA, 2000.

[5] C.-J. Chiang and G.S. Chirikjian. Modular robot motion planning using similarity metrics. *Autonomous Robots*, 10(1):91–106, 2001.

[6] G. Chirikjian, A. Pamecha, and I. Ebert-Uphoff. Evaluating efficiency of self-reconfiguration in a class of modular robots. *Journal of Robotics Systems*, 13:317–338, 1996.

[7] G.S. Chirikjian. Metamorphic hyper-redundant manupulators. In *Proc. of the 1993 JSME Int. Conf. on Advanced Mechatronics*, pages 467–472, Tokyo, Japan, 1993.

[8] K. Hosokawa, T. Tsujimori, T. Fujii, H. Kaetsu, H. Asama, Y. Kuroda, and I. Endo. Self-organizing collective robots with morphogenesis in a vertical plane. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 2858–2863, Leuven, Belgium, 1998.

[9] B. Khoshnevis, B. Kovac, W.-M. Shen, and P. Will. Reconnectable joints for self-reconfigurable robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Maui, Hawaii, USA, 2001.

[10] K. Kotay, D. Rus, M. Vona, and C. McGray. The self-reconfiguring robotic molecule. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 424–431, Leuven, Belgium, 1998.

[11] S. Murata, H. Kurokawa, and S. Kokaji. Self-assembling machine. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 441–448, San Diego, USA, 1994.

[12] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, and S. Kokaji. A 3-d self-reconfigurable structure. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 432–439, Leuven, Belgium, 1998.

[13] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, and S. Kokaji. Hardware design of modular robotic system. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2210–2217, Takamatsu, Japan, 2000.

[14] A. Pamecha, C. Chiang, D. Stein, and G.S. Chirikjian. Design and implementation of metamorphic robots. In *Proc. of the ASME Design Engineering Technical Conf. and Computers in Engineering Conf.*, pages 1–10, Irvine, USA, 1996.

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

[16] W.-M. Shen, B. Salemi, and P. Will. Hormone-based control for self-reconfigurable robots. In *Proc. of the Int. Conf. on Autonomous Agents*, pages 1–8, Barcelona, Spain, 2000.

[17] W.-M. Shen, B. Salemi, and P. Will. Hormones for self-reconfigurable robots. In *Proc. of the Int. Conf. on Intelligent Autonomous Systems*, pages 918–925, Venice, Italy, 2000.

[18] K. Støy, W.-M. Shen, and P. Will. Global locomotion from local interaction in self-reconfigurable robots. In *Proc. of the 7th Int. Conf. on Intelligent Autonomous Systems (IAS-7)*, pages 309–316, Marina del Rey, California, USA, 2002.

[19] K. Støy, W.-M. Shen, and P. Will. How to make a self-reconfigurable robot run. In *Proc. of the First Int. Joint Conf. on Autonomous Agents & Multiagent Systems*, pages 813–820, Bologna, Italy, 2002.

[20] K. Støy, W.-M. Shen, and P. Will. On the use of sensors in self-reconfigurable robots. In *Proc. of the Seventh Int. Conf. on The Simulation of Adaptive behavior (SAB'02)*, pages 48–57, Edinburgh, UK, 2002.

[21] K. Tomita, S. Murata, H. Kurokawa, E. Yoshida, and S. Kokaji. A self-assembly and self-repair method for a distributed mechanical system. *IEEE Transactions on Robotics and Automation*, 1999.

[22] C. Ünsal and P.K. Khosla. Mechatronic design of a modular self-reconfiguring robotic system. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 1742–1747, San Francisco, USA, 2000.

[23] M. Yim. *Locomotion with a unit-modular reconfigurable robot*. PhD thesis, Department of Mechanical Engineering, Stanford University, 1994.

[24] M. Yim. New locomotion gaits. In *Proc. of Int. Conf. on Robotics & Automation*, pages 2508 –2514, San Diego, California, USA, 1994.

[25] M. Yim, D.G. Duff, and K.D. Roufas. Polybot: A modular reconfigurable robot. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 514–520, San Francisco, USA, 2000.