Using Role-Based Control to Produce Locomotion in Chain-Type Self-Reconfigurable Robots

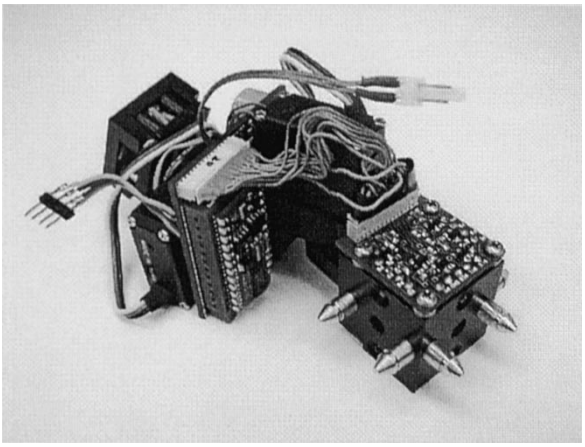
Kasper Støy, Wei-Min Shen, and Peter M. Will

***Abstract*—This paper presents a role-based approach to the problem of controlling locomotion of chain-type self-reconfigurable robots. In role-based control, all modules are controlled by identical controllers. Each controller consists of several playable roles and a role-selection mechanism. A role represents the motion of a module and how it synchronizes with connected modules. A controller selects which role to play depending on the local configuration of the module and the roles being played by connected modules. We use role-based control to implement a sidewinder and a caterpillar gait in the CONRO self-reconfigurable robot. The robot is made from up to nine modules connected in a chain. We show that the locomotion speed of the caterpillar gait is constant even with loss of 75% of the communication signals. Furthermore, we show that the speed of the caterpillar gait decreases gracefully with a decreased number of modules. We also implement a quadruped gait and show that without changing the controller the robot can be extended with an extra pair of legs and produce a hexapod gait. Based on these experiments, we conclude that role-based control is robust to signal loss, scales with an increased number of modules, and is a simple approach to the control of locomotion of chain-type self-reconfigurable robots**

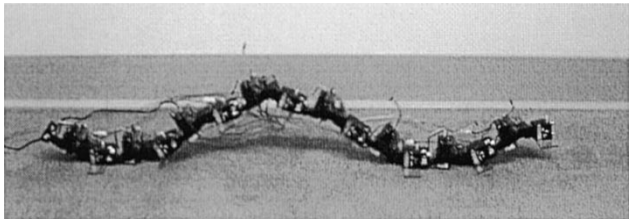
***Index Terms*—Control, locomotion, self-reconfigurable robots.**

I. Introduction

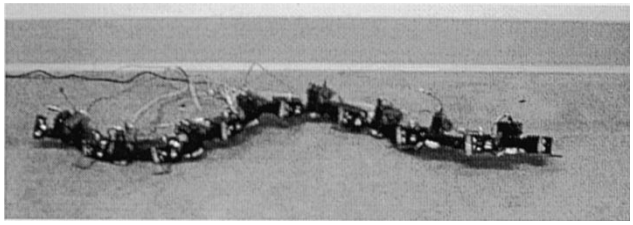
IMPORTANT performance characteristics for robots in controlled production environments are precision, efficiency, and repeatability. However, the importance of these characteristics decreases and characteristics such as robustness, versatility, and adaptability become important when moving the robots out of the controlled production environments and into dynamic real world task-environments. These task-environments include missions on other planets, search and rescue in collapsed buildings, and battle field reconnaissance. Robustness is needed because the system cannot rely on a human operator for repair. In some tasks it is even impossible for a human to assist the robot. Versatility is important because the robot should be able to solve unexpected problems which arise on its way to complete its task. It is also important that the robot is able to adapt-not only to complete an unexpected task, but also to be able to handle tasks which are similar in terms of result but are different in terms of the conditions under which they are performed.



(a)



(b)



(c)

Fig. 1. (a) CONRO module. (b) CONRO robot performing caterpillar-like locomotion. (c) A gait similar to that of a sidewinder snake. The cables only supply power.

Self-reconfigurable robots may offer a platform with these characteristics. A self-reconfigurable robot is a robot built from potentially many independent modules that connect to form a robot. These modules can have their own power supply, actuators, sensors, communication system, and computational capabilities; see the *CON*figurable *RO*bot (CONRO) module in Fig. 1 or refer to the physical realizations described in [8]–[10],

Manuscript received March 31, 2002; revised October 1, 2002. The work reported here was performed while visiting the Information Sciences Institute, University of Southern California, Los Angeles, CA. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract DAAN02-98-C-4032, in part by Air Force Office of Scientific Research (FOSR) under Contract F49620-01-1-0020, in part by the Eurpopean Union (EU) under Contract IST-20001-33060, and in part by the Danish Technical Research Council under Contract 26-01-0088. Recommended by Guest Editors W.-M. Shen and M. Yim.

K. Støy is with the Adaptronics Group, The Maersk Mc-Kinney Moller Institute for Production Technology, University of Southern Denmark, Odense, DK-5230, Denmark (e-mail: kaspers@mip.sdu.dk).

W.-M. Shen and P. M. Will are with the Information Sciences Institute, University of Southern California, Los Angeles, CA 90089 USA (e-mail: shen@isi.edu; will@isi.edu).

Digital Object Identifier 10.1109/TMECH.2002.806223

[12], [14], [19], [23]. Self-reconfigurable robots gain their robustness through redundancy. The robot is built from many identical modules and, therefore, if one fails it can be replaced by a spare module. The modules can be connected in different ways making the same robotic system able to perform a wide range of tasks. Due to this, self-reconfigurable robots can achieve levels of versatility superior to that of traditional fixed-shape robots. Self-reconfigurable robots can also use this ability to adapt to the task. Finally, since these robots are built from many identical modules these can be mass-produced to keep their cost low compared to their complexity. In order to realize these promising potentials many research challenges have to be met. One of these is to understand how to control locomotion of self-reconfigurable robots which is the main topic of this paper.

The paper is organized as follows. In Section II we review related work on locomotion of self-reconfigurable robots. We note that there exist two general approaches: synchronous control and asynchronous control. In synchronous control the modules are synchronized before each action in order to guarantee that the desired global behavior is achieved. In asynchronous control this idea has been abandoned for reasons of robustness and scalability. Instead each module acts independently based on local information. The idea is that the desired global behavior will emerge from these local interactions. However, using this approach it is hard to generate globally coordinated locomotion.

In Section III, we argue that it might not be desirable to make the synchronization mechanism independent of the actions of the modules, because in this case an additional mechanism is needed to coordinate which module does what and in what sequence. Therefore, we combine a synchronization mechanism based on local communication and local timers with the actions of the module and name this a *role*. We use these roles to control the robot and call it *role-based control* (note: this should not be confused with rule-based control). We then describe how role-based control can be used to make a chain of eight CONRO modules produce locomotion similar to that of a caterpillar and a sidewinder snake. We show that these systems work independent of the number of modules in the system and are robust to communication failures.

In Section VIII, we argue that modules should be able to play different role. We show how modules can select roles based on the local configuration and the roles being played by connecting modules. We use this to implement a quadruped and hexapod walking gait in the CONRO self-reconfigurable robot. We show scalability properties of this system. In Section X we discuss remaining issues and finally arrive at the conclusion in Section XI.

II. Related Work

In locomotion of self-reconfigurable robots, there exist two broad classes of algorithms. Hosakawa *et al.* [6] and Butler *et al.* [2] introduced the "water-flow" algorithms where modules based on local rules travel over each other to produce locomotion. In this class of algorithms the robot locomotes by changing configuration. Another approach, the one we will investigate in this paper, is to achieve locomotion by controlling joint positions of the modules of the robot. This has earlier been investigated by Yim [21], [22] and Shen, Salemi, and others [13]–[15].

In Yim's work, the locomotion gait is represented in a gait-control table. Each column in the table represents a sequence of actions that one module has to perform. The modules make a transition from one row to the next row based on a synchronization signal from a central host. This implies that the system has a single point of failure. If the host fails the entire system fails. Furthermore, if a module does not receive a synchronization signal it will have severe effects on the locomotion pattern. As a solution to this problem, it is proposed that modules can use timers to stay synchronized, but as we will discus below timers drift. Another problem is that the modules need to know which column in the gait control table to execute. This means that the modules need IDs and can, therefore, not be interchanged or added unless the table is rewritten. This also implies that the system cannot handle module failures. Yim has shown in his work that the approach is versatile, scales and the complexity is manageable in system consisting of tens of modules.

Shen, Salemi, and others use a hormone that travels through all the modules to synchronize the action execution of the modules. This means that the performance of the system scales linearly with the number of modules. Hormone based control avoids the problems of having a central host sending out synchronization signals. However, the system stops responding if a hormone is lost somewhere for instance due to a module failure. This means that the system is not robust in this respect, but the system is robust to reconfiguration. Hormone based control have the potential to be a versatile control method. However, the complexity of using this method in more complex systems may become increasingly difficult, but this is a question for further research.

III. Synchronization

In the previous section we outlined two broad categories of control systems: systems where the modules are synchronized before each actions (gait control tables and hormone based systems) and those where they do not need to synchronize (the water-flow algorithms). We acknowledge that the modules need to be autonomous in other to ensure robustness and scalability. However, we also acknowledge that in some tasks it is useful to keep the modules synchronized.

We propose that the useful properties of these two categories can be combined. The first step toward achieving this is to allow synchronization over time. If modules keep track of time using local timers and from time to time synchronize their timers with connected modules, the entire system will eventually be synchronized. This assumes that modules can stay synchronized without communicating for some time. This is achievable, because most processors are equipped with timers. These times might be too imprecise to keep modules synchronized infinitely, but are good enough to keep the modules synchronized for a while.

In order to illustrate this point we made an experiment where four CONRO modules run identical programs. Each controller makes one of the module's degrees of freedom perform an oscillation with period $T = 2.37$ s. We then started these modules at

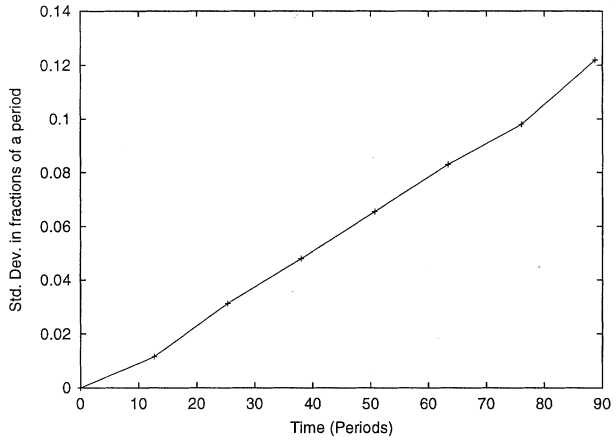


Fig. 2. This graph shows how four modules performing the same movement lose synchronization over time. The modules started at the same time making an oscillation with period $T = 2.37$ s. The x axis is the number of periods. The y axis is the standard deviation of the position of the modules measured in fractions of a period. For instance, after 90 periods the modules have drifted apart on average more than 1/10 of a period.

the same time and observed how their synchronization degraded over time. In Fig. 2 it can be seen how the standard deviation of the position of the modules increases over time. It can be seen how often a synchronization signal is needed depends on the precision needed in the system. For instance, it is questionable if a drift of 1% has any impact on the system. If 1% drift can be accepted we only need to synchronize every 10th period.

In a synchronized system, all modules have to communicate before each action because this is the only way a module can find out when to perform an action. However, with the introduction of timers we can make the optimistic assumption that the timer can be trusted for synchronization. This implies that not all modules need to communicate at each time step to stay synchronized.

We can use one of the many algorithms for synchronizing clocks (see [11], [16] for an overview) and achieve increased efficiency because the local timer can be relied on for deciding when to perform a new action. However, this is not the path we will follow here. In order to achieve coordinated global behavior of the robot it is important that modules act at the right time which is not necessarily at the same time. Therefore, we combine the synchronization mechanism with the actions of the modules in role-based control that we will describe in Section IV.

IV. CONRO Modules

Before we go on to describe role-based control we will describe the CONRO self-reconfigurable robot. The CONRO modules were developed at University of Southern California's Information Sciences Institute [3], [7] (see Fig. 1). The modules are roughly shaped as rectangular boxes measuring 10 cm $\times$ 4.5 cm $\times$ 4.5 cm and weigh 100 g. The modules have a female connector located at one end by definition facing south and three male connectors located at the other end facing east, west, and north. 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.[1]

V. Role-Based Control

We assume that a parent connector is specified and the remaining connectors are defined to be child connectors. A module can only attach to a parent module by attaching its parent connector to one of the parent's child connectors. For instance, on the CONRO module we specify the female connector as the parent connector and the three male connectors are then the child connectors. This implies that no matter how the CONRO robot is configured the physical characteristics of the connectors make sure this assumption holds.

Using this assumption it is possible to make at most one loop in the configuration. This is undesirable for reasons we will discus below so we assume that there are no loops in the configuration. These two assumptions combined imply that we are limited to tree configurations with well-defined parent child relationships. We will later see that these assumptions lead to many simplifications. Note that these assumptions make role-based control suitable for any chain-type self-reconfigurable robot.

Fundamental to role-based control is the notion of a role. A role consists of three components. A cyclic action sequence $A(t)$ where t is an integer and $t \in [0 : T]$. T is the period of the action sequence and the second parameter that needs to be specified. In our implementation $A(t)$ just returns the joint positions of the module for each t. However, it is also possible to extend $A(t)$ to include feedback from the environment to bias the motion. The important characteristic is that the resulting motion is cyclic. For instance, if a module acts as a leg it can move in order to step over obstacles as long as it supports the weight of the robot when the remaining modules expect it to do so. The third parameter is a set of delays D. A delay $d_i \in D$ specifies the time t at which a synchronization signal is to be sent to the module connected to child connector i. The following shows the caterpillar role as an example:

$$A(t, \text{caterpillar}) = \begin{cases} \text{pitch}(t) = 50^\circ \sin\left(\frac{2\pi}{T}t\right) \\ \text{yaw}(t) = 0 \end{cases}$$
$$d_{\text{north}} = \frac{T}{5}$$
$$T = 180.$$

The modules play a role using the algorithm outlined in Fig. 3. The algorithm starts by setting $t = 0$ and continues to the main loop. Here, the algorithm first checks if t is equal to the delay specified for each child connector. In case t equals one of these delays d_i a synchronization signal is sent through the corresponding child connector i. If the module has received a signal from its parent, t is reset. After that the joints are moved to the position described by $A(t)$. t is incremented unless a period has been completed in which case t is reset. Finally, another iteration of the loop is initiated.

[1]Refer to http://www.isi.edu/conro for more details and for videos of the experiments reported in this paper.

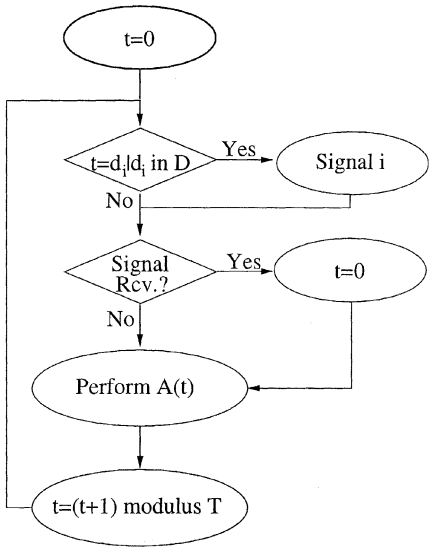


Fig. 3. Visualization of the role playing part of the algorithm. See Section V for an explanation.

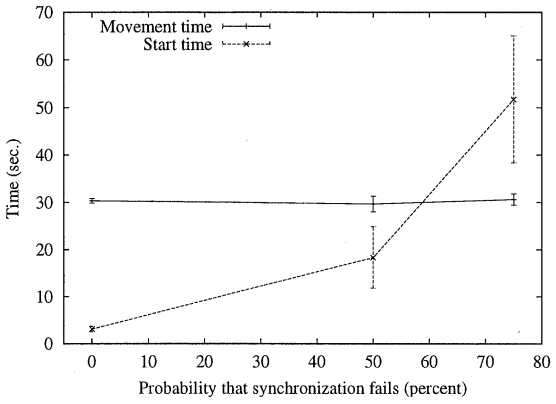


Fig. 4. This figure shows how long it takes for the robot to start and to move 87 cm as a function of the chance of losing a synchronization signal. Each data point represents the average and standard deviation of ten experiments.

We can now see the reason why we have to assume that there are no loops in the configuration: there is a risk that synchronization signals chase each other around in the loop and, thus, the system may never converge. We have pointed out a potential solution to this problem in [17].

VI. Caterpillar

First, we will present the implementation of a caterpillar-like locomotion gait in the CONRO robot. We will demonstrate the properties of the implementation in term of robustness to loss of communication signals and reconfiguration. Caterpillar-like locomotion has in the context of self-reconfigurable robots previously been studied in [5], [14], [20].

A. Implementation

We configured eight CONRO modules in a chain and downloaded the caterpillar role described above into each of them. The first module of the chain is then started using an external infra-red signal. The first module starts its action sequence. After 1/5th of a period (d_{north}) a synchronization signal is sent to the child module connected to the northern child connector. If the signal is received the second module starts to move and after another 1/5th of a period a synchronization signal is sent to the third module and so on. The modules repeatedly send a synchronization signal each period. Therefore, the system is not sensitive to loss of synchronization signals. When all modules have been started and are synchronized to be 1/5th of a period apart they produce a traveling sine wave resulting in the caterpillar-like locomotion gait shown in Fig. 1. The robot moves at a speed of 2.9 cm/s. We will refer to the robot in a chain configuration playing the caterpillar role as the caterpillar.

B. Experiments—Robustness to Loss of Communication Signal

In the first series of experiments, we want to examine the robustness of the system to communication errors. In order to do this we introduce artificial communication errors by randomly deciding if a signal is sent or not. We did three separate series of experiments where the chance of a signal being sent is respectively 25%, 50%, and 100%. We want to fint out what impact these communication errors have on the time it takes for the system to synchronize initially and the locomotion speed of the robot.

The robot is started using an external infra-red signal and it is recorded how long it takes for all modules to start. When all the modules have started we record how long it takes for the robot to move a distance of 87 cm. We repeated these experiments ten times for each level of signal loss. The results of these experiments can be seen in Fig. 4.

It can be observed from Fig. 4 that as the probability of signal loss increases the start time increases. This was expected, because if a synchronization signal is lost it takes a full period before the next synchronization signal is sent. However, the figure also shows that the locomotion time stays constant. In fact Student's t-test shows at the 5% confidence level that the hypothesis that the times can be assumed to equal in the three experiments is accepted (see appendix for details). This extreme robustness to communication error is not surprising, because Fig. 2 showed that the modules can stay synchronized for a significant number of periods before they drift so much apart that the effect is measurable. In fact the data shown in Fig. 2 is produced by four unconnected modules playing the caterpillar role.

We made another series of experiments designed to show how the performance of the system changes with the number of modules. We made experiments with caterpillars made of 2, 4, and 8 modules. We measured how long it took them to move 87 cm. The results of these experiments are visualized in Fig. 5. The figure shows that even though the speed decreases with the number of modules the system still works. The performance decreases because the caterpillar role is designed to be used by a chain of more than four modules. In order to produce efficient locomotion the caterpillar needs at least two contact points with the ground at all times. In the caterpillar role, the sine wave repeats itself after four modules. This means that most of the time the chain with four modules only have one contact point resulting in less than optimal speed.

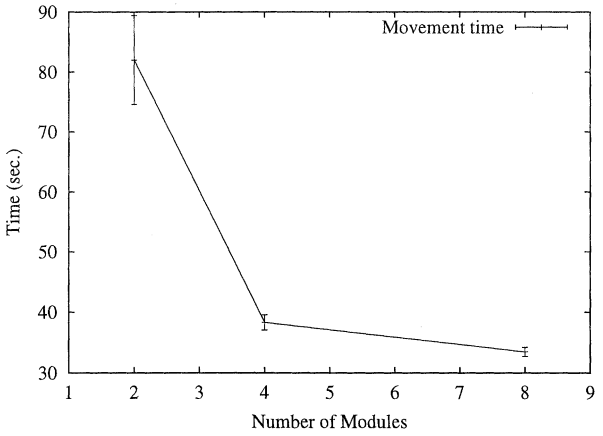


Fig. 5. This figure shows how long it takes for the caterpillar to move 87 cm as a function of the number of modules connected. The data points for 4 and 8 modules represent the average and standard deviation of ten experiments. The data point for 2 modules represents five experiment.

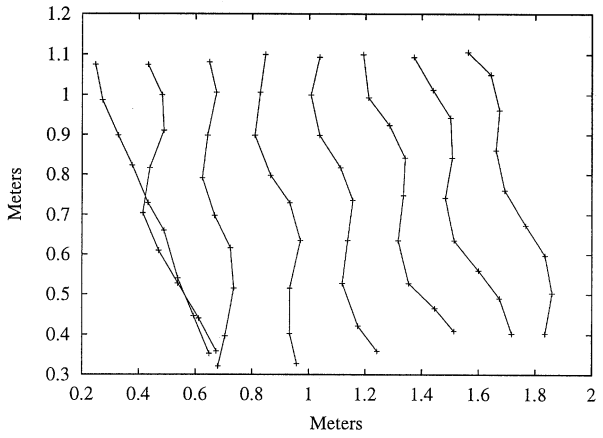


Fig. 6. We recorded the locomotion of the sidewinder using an overhead camera. This figure shows the position of the modules of the sidewinder recorded every 3 s.

VII. Sidewinder

When first the idea of role-based control is at hand it is easy to develop more locomotion gaits. We will now present an implementation of a gait similar to that of a sidewinder snake. This gait has been analyzed in detail by Burdick *et al.* [1], but here we just use the intuition that if segments of the body moving in one direction are lifted from the ground and segments moving in the other touch the ground locomotion is produced. The sidewinder role looks as follows:

$$A(t, \text{sidewinder}) = \begin{cases} \text{pitch}(t) = 20^\circ \cos\left(\frac{2\pi}{T}t\right) \\ \text{yaw}(t) = 50^\circ \sin\left(\frac{2\pi}{T}t\right) \end{cases}$$
$$d_{\text{north}} = \frac{T}{5}$$
$$T = 180.$$

When all the modules are connected in a chain and synchronized it looks as shown in Fig. 1. We refer to this as the sidewinder. We recorded the movement of the sidewinder using an overhead camera. We then recorded the position of the modules every three seconds. The result of this analysis can be seen in Fig. 6. We repeated this experiment five times with similar results. The sidewinder moves at a speed of 6.7 cm/s.

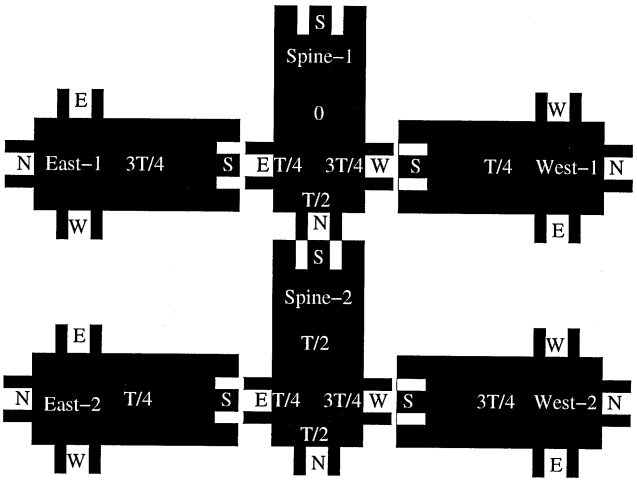


Fig. 7. Black boxes represent modules. The modules are connected to form a quadruped robot. All the modules are named and labeled with compass directions. The expression next to a connector represents the delay d across that connector. The delay is expressed in fractions of a period T. For instance, the delay associated with the east connector of the spine module is $d_{\text{east}} = T/4$. The expressions located in the center of the modules represent the value of t of that module when the spine module spine-1 is at $t_{\text{spine}-1} = 0$. The t value of a child is calculated using $t_{\text{child}-\text{i}} = t_{\text{parent}} - d_i$ where i is the connector to which the child is connected. Note using the delays specified here the top east leg and bottom west legs are synchronized. This also goes for the top west and bottom east leg.

VIII. Multiple Roles

Insisting that all modules play the same role limits the number of applications of role-based control, therefore, we want modules to be able to play different role. This raises the question of role selection. How does a module decide which role to play? In role-based control the role selection is based on information about the local configuration and the roles being played by connected neighboring modules. It is also possible to extend role-based control to select roles based on sensor feedback from the environment which we have studied in [18] or timing information.

The role being played r can be a function of the parent's role r_p, the child connector of the parent that the module is connected to c_p, the set of connectors where child modules are attached C, the roles of the connected child modules r_i where $i \in C$, and the current role r being played by the module

$$r = f\left(r_p, c_p, C, r_i, r\right), \qquad \text{where } i \in C.$$

This is a general function and rarely the role of a module will depend on all parameters. For instance, a module might select its role based only on its local configuration given by c_p and C. One important point is that since we only consider tree configurations the root of the tree can be uniquely found: the module with no parent. The root module can play a root role. This enables the children of the root to uniquely find their position in the configuration tree through the parent's role r_p and the connector of the parent to which the child is connected c_p. By induction we can see that if it is needed it is possible for all modules in the tree to discover their position in the configuration tree. Again,

```
r = <start role>
t = 0
while (1)
  if (t=d(r)_1) then
    <send message M(r,1) to child connector 1>
    <update r>
  endif
  ...
  if (t=d(r)_n) then
    <send message M(r,n) to child connector n>
    <update r>
  endif

  if <message m received from parent connec-
tor> then
    t=0
    <update r based on m>
  endif

  <perform action A(r,t)>
  t = (t+1) modulus T(r)
endwhile
```

Fig. 8. Role playing algorithm with role selection.

often this is not desirable, because if information can be kept as local as possible the system can adapt faster.

All modules have the same role-selection mechanism and exactly what role a module will play is decided based on f. This implies that there is no centralized representation of the desired global configuration and the corresponding roles. Therefore, if a configuration sub tree is cutoff the root of this sub-tree can quickly discover it and trigger appropriate role changes further down in the sub-tree. At the same time the parent tree can discover that the sub-tree was cutoff and trigger role changes up in the tree. This means that this role selection system is robust to reconfiguration.

In our current implementation, the role and configuration information is contained in the synchronization signal. Therefore, the role-selection mechanism also preserves the robustness to loss of communication signal of role-based control. If a signal is lost the role and configuration information will be propagated a period later. In Fig. 8, it can be seen how the role playing algorithm with the added role-selection mechanism looks.

IX. Walkers

In order to verify the usability of this role-selection mechanism we implemented a walking gait in the CONRO self-reconfigurable robot.

A. Role Definitions

We configured the robot in a quadruped configuration as shown in Fig. 9(a). In this configuration three different roles are need to make the robot walk: a spine role, a left leg, and right leg.

The modules making up the spine should play the spine role. The spine role makes the spine oscillate to increase the step length. Furthermore, the delays of the spine makes sure that the rear legs are half a period delayed compared to the front legs (see Fig. 7 for details)

$$A(t, \text{spine}) = \begin{cases} \text{pitch}(t) = 0^\circ \\ \text{yaw}(t) = 25^\circ \cos\left(\frac{2\pi}{T}t + \pi\right) \end{cases}$$

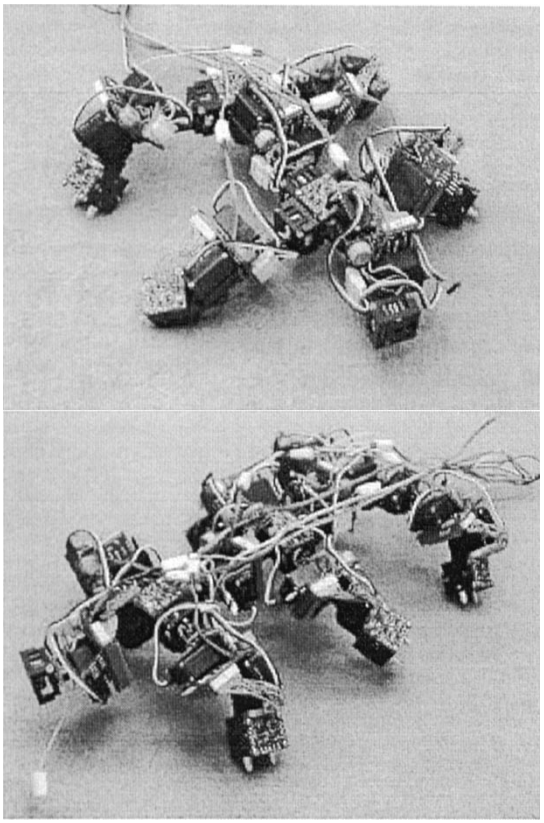


Fig. 9. CONRO robot configured as a quadruped and hexapod walker.

$$\begin{aligned} d_{\text{east}} &= \frac{T}{4} \\ d_{\text{south}} &= \frac{2T}{4} \\ d_{\text{west}} &= \frac{3T}{4} \\ T &= 180. \end{aligned}$$

The east legs play the east leg role below and the west legs play the same role except t is replaced by $2\pi - t$. This gives the same motion, but in the opposite direction

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

In order to combine these roles into a working control system we need to define when a module should play what role. We make the following rules: a module plays a spine role if it has child modules connected to the east and west child connector. A module plays east leg if it is connected to the east connector of the parent module and a similar rule works for the west leg

$$r = \begin{cases} \text{spine}, & \text{if west, east} \in C \\ \text{west leg}, & \text{if } c_p = \text{west} \\ \text{east leg}, & \text{if } c_p = \text{east}. \end{cases}$$

B. Experiments—Scalability

In order to test this implementation, we measured how long it took the CONRO robot in a quadruped configuration to cover a distance of 150 cm. We found that the average of ten trials was

10.9 s and the standard deviation was 0.57 s. This corresponds to a speed of 13.8 cm/s.

We now added an extra pair of legs. Note that this does not require any changes to the controller, because the synchronization mechanism already implemented will make sure that the third pair of legs are half a period delayed compared to the second pair of legs. We repeated the experiments another ten times and found that the average time was 12.0 s (12.5 cm/s) and the standard deviation was 0.57 s.

Initially we tested the hypothesis that the speed of the robot is independent of the number of modules. Unfortunately, Student's t-test rejected this hypothesis. From close observation of the experiments we found that the quadruped robot takes a longer step, because it slides a little forward with each step due to its momentum. In the hexapod this is not the case, because of the friction caused by the extra pair of legs. In order to remove this difference from our data we returned to the videos of the experiments and counted the number of steps taken by the robot in each experiment. We divided the time with the number of steps to produce a time per step measure. We then tested the hypothesis that the time per step is the same for both the quadruped and hexapod walker. This hypothesis was accepted on the 5% confidence level (see the appendix). This implies that the speed of the system is identical in the two experiments. This finding and the fact that role-based control uses constant time to keep the system synchronized leads us to conclude that it is likely that role-based control scales.

X. Discussion

In role-based control, one of the basic assumptions is that the modules form a tree configuration. If we assume that the modules form a loop, we introduce the problem that there is no unique leader that emerges in a similar way to the root in a configuration tree. This problem reduces to the well-known problem of leader election in a ring to which many solutions exists for instance [4]. When the leader has been found, the leader should ignore any messages it receives from its parent and as a result the loop is removed from a software point of view. We have used this idea to make rolling track locomotion in [17]. However, it is not straightforward how to generalize role-based control to work in a general configuration without losing the simplicity and robustness of the approach.

We have shown through the experiments that role-based control is likely to scale, is robust to signal loss, and reconfiguration. We have shown through the introduction of role selection, how it is possible to give the same control system versatility in terms of what tasks the system can perform. Role-based control also holds the potential to be robust to module failure, because if a module fails it can just be ejected and the two remaining parts of the tree can continue to function on their own.

In this paper, we have not focused on the problem of how to design the roles for a more complex system. In the systems we have presented here, the complexity is manageable for humans. However, when the number of modules increase it might be increasingly difficult for a human designer to handle the complexity. This problem is also inherent in the other methods used for locomotion of self-reconfigurable robots.

TABLE I

F-test	50%	25%
100%	0.0027	0.025
50%	-	0.32

T-test	50%	25%
100%	0.28	0.52
50%	-	0.17

TABLE II

F-test	1
t-test	$3.0 * 10^{-7}$

F-test	0.37
t-test	0.78

We mentioned earlier that the action sequences of roles can be biased by external sensing and roles might also be changed in response to this sensor information. In [18], we have shown how it is possible for the quadruped walker to use sensors to avoid obstacles. However, a full understanding of this problem is also an issue to be addressed in the future.

XI. Conclusion

In this paper, we have presented role-based control as an approach to controlling locomotion of chain-type self-reconfigurable robots. We have, through our presentation and experiments, argued that role-based control is a robust and versatile approach to the control of locomotion in self-reconfigurable robots. We have furthermore pointed out that it is likely that it scales with an increased number of modules.

Appendix

In the statistical tests, we use an F-test to test the hypothesis that the two data sets have equal variance. Depending on this result, we pick the appropriate version of Student's t-test to test if the hypothesis that the mean value of the two data sets are equal.

The first three data sets are from the caterpillar experiment where we test the hypothesis that the caterpillar takes the same time to move 87cm independent of the chance of signal loss. Table I shows that at the 5% level the hypothesis is accepted and, thus, it can be assumed that it takes the same time for the caterpillar to move independent of the chance of loss of signal. The next series of data is from the walker experiments. First we test the hypothesis that the time used to walk 150 cm is the same for the quadruped and the hexapod walker. As shown in the left-hand side of Table II, this hypothesis is rejected. Second, we test the hypothesis that the time per step is equal in the quadruped and hexapod walker. This hypothesis is accepted and, therefore, we conclude that the performance of the two systems is independent of the number of modules.

References

[1] J. W. Burdick, J. Radford, and G. S. Chirikjian, "A sidewinding locomotion gait for hyper-redundant robots," *Adv. Robot.*, vol. 13, no. 4, pp. 531–545, 1995.

[2] Z. Butler, K. Kotay, D. Rus, and K. Tomita, "Generic decentralized control for a class of self-reconfigurable robots," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Washington, DC, 2002, pp. 809–815.

[3] A. Castano, R. Chokkalingam, and P. Will, "Autonomous and self-sufficient conro modules for reconfigurable robots," in *Proc. 5th Int. Symp. Distributed Autonomous Robotic Systems*, Knoxville, TX, 2000, pp. 155–164.
[4] E. Chang and R. Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processes," *Communicat. ACM*, vol. 22, no. 5, pp. 281–283, 1979.
[5] G. S. Chirikjian and J. W. Burdick, "The kinematics of hyper-redundant robot locomotion," *IEEE Trans. Robot. Automat.*, vol. 11, pp. 781–793, Dec. 1995.
[6] 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. IEEE Int. Conf. Robotics and Automation*, Leuven, Belgium, 1998, pp. 2858–2863.
[7] B. Khoshnevis, B. Kovac, W.-M. Shen, and P. Will, "Reconnectable joints for self-reconfigurable robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Maui, HI, 2001.
[8] K. Kotay, D. Rus, M. Vona, and C. McGray, "The self-reconfiguring robotic molecule," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, 1998, pp. 424–431.
[9] S. Murata, E. Yoshida, H. Kurokawa, K. Tomita, and S. Kokaji, "Self-repairing mechanical systems," *Autonomous Robots*, vol. 10, no. 1, pp. 7–21, 2001.
[10] A. Pamecha, C. Chiang, D. Stein, and G. S. Chirikjian, "Design and implementation of metamorphic robots," in *Proc. ASME Design Engineering Technical Conf. and Computers in Engineering Conf.*, Irvine, CA, 1996, pp. 1–10.
[11] P. Ramanathan, K. Shin, and R. Butler, "Fault-tolerant clock synchronization in distributed systems," *IEEE Trans. Comput.*, vol. 23, pp. 33–44, Oct. 1990.
[12] D. Rus and M. Vona, "A physical implementation of the crystalline robot," in *Proc. of IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, 2000, pp. 1726–1733.
[13] B. Salemi, W. Shen, and P. Will, "Hormone controlled metamorphic robots," in *Proc. EEE Int. Conf. Robotics and Automation*, Seoul, Korea, 2001, pp. 4194–4199.
[14] W.-M. Shen, B. Salemi, and P. Will, "Hormone-based control for self-reconfigurable robots," in *Proc. Int. Conf. Autonomous Agents*, Barcelona, Spain, 2000, pp. 1–8.
[15] ——, "Hormones for self-reconfigurable robots," in *Proc. Int. Conf. Intelligent Autonomous Systems*, Venice, Italy, 2000, pp. 918–925.
[16] B. Simons, J. L. Welch, and N. Lynch, "An overview of clock synchronization," in *Proc. Asilomar Workshop on Fault-Tolerant Distributed Computing Conf.*, vol. 448, 1990, pp. 84–96.
[17] K. Støy, W.-M. Shen, and P. Will, "Global locomotion from local interaction in self-reconfigurable robots," in *Proc. 7th Int. Conf. Intelligent Autonomous Systems (IAS-7)*, Marina del Rey, CA, 2002, pp. 309–316.
[18] ——, "On the use of sensors in self-reconfigurable robots," in *Proc. 7th Int. Conf. Simulation of Adaptive Behavior (SAB'02)*, Edinburgh, U.K., 2002, pp. 48–57.
[19] C. Ünsal and P. K. Khosla, "Mechatronic design of a modular self-reconfiguring robotic system," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, 2000, pp. 1742–1747.
[20] M. Yim, "A reconfigurable modular robot with many modes of locomotion," in *Proc. JSME Int. Conf. Advanced Mechatronics*, Tokyo, Japan, 1993, pp. 283–288.
[21] ——, "Locomotion with a unit-modular reconfigurable robot," Ph.D. dissertation, Dept. Mech. Eng., Stanford Univ., 1994.
[22] ——, "New locomotion gaits," in *Proc. Int. Conf. Robotics and Automation*, San Diego, CA, 1994, pp. 2508–2514.
[23] M. Yim, D. G. Duff, and K. D. Roufas, "Polybot: A modular reconfigurable robot," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, 2000, pp. 514–520.

Kasper Støy received the M.S. degree in computer science from the University of Aarhus, Aarhus, Denmark, in 1999. He is currently working toward the Ph.D. degree at The Maersk Institute for Production Technology, University of Southern Denmark, Odense, Denmark. As part of his Ph.D. program he visited the Information Sciences Institute, University of Southern California, Los Angeles, where the research presented in this paper was performed.

Previously, he was a Research Scientist at the Robotics Laboratories, University of Southern California, where he conducted research on biology inspired multirobot coordination. His research interests include self-reconfigurable robots, biological inspired multirobot coordination, and robot learning.

Wei-Min Shen received the Ph.D. degree from Carnegie-Mellon University, Pittsburgh, PA, in 1989, under Prof. Herbert Simon.

Currently, he is the Director of the Polymorphic Robotics Laboratory, Information Science Institute (ISI), the Associate Director of the Center for Robotics and Embedded Systems, and a Research Assistant Professor of computer science all at the University of Southern California (USC), Los Angeles. His research is sponsored by the National Science Foundation (NSF), Air Force Office of Scientific Research (AFOSR), Defense Advanced Research Projects Agengy (DARPA), and the National Aeronautics and Space Administration (NASA). He is the author of the book, "*Autonomous Learning from the Environment*" (New York: W.H. Freeman, 1994). His achievements have been reported by the news media, including CNN, PBS, the Discovery channel, *The LA Times*, *BYTE*, the *Chinese World Journal*, and *SCIENCES*. He has Chaired several international conferences and workshops in robotics, machine learning, and dating mining and served on the Editorial Boards for two scientific books and one international journal. His current research interests include self-reconfigurable robots, artificial intelligence, and machine learning.

Dr. Shen is the recipient of a 1996 AAAI Robotics Competition Silver-Medal Award, a 1997 RoboCup World Championship Award, and a 1997 Meritorious Service Award at USC/ISI.

Peter M. Will received the B.Sc. degree in electrical engineering and the Ph.D. degree in nonlinear control systems from the University of Aberdeen, Aberdeen, Scotland, U.K., in 1958 and 1960, respectively.

Previously, he spent 16 years at the IBM T. J. Watson Research Laboratories, Yorktown, NY, seven years with Schlumberger, Houston, TX, five years at Hewlett-Packard Labs, Palo Alto, CA, and six years as a member of the Information Science and Technology (ISAT) Group working with the Defense Advanced Research Projects Agency (DARPA). Currently, he is the Director of the Distributed Scalable Systems Division at the Information Sciences Institute, University of Southern California (USC), Marina Del Rey, and a Research Professor of industrial and systems engineering at the same university. He has over 50 publications and 10 patents.

Dr. Will was awarded the International Engelberger Prize in robotics in 1990.