Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea · May 21-26, 2001

Hormone-Controlled Metamorphic Robots

Behnam Salemi, Wei-Min Shen, and **Peter Will**
USC Information Science Institute and Computer Science Department
4676 Admiralty Way, Marina del Rey, Ca. 90292, USA
{salemi,shen,will}@isi.edu

1 Abstract

Metamorphic robots with shape-changing capabilities provide a powerful and flexible approach to complex tasks in unstructured environments. However, due to their dynamic topology and decentralized configuration, metamorphic robots demand control mechanisms that go beyond those used by conventional robots. This paper builds on our previous results of hormone-based control, and develops a novel distributed control algorithm called CELL that can select, synchronize, and execute gaits and other reconfiguration actions without assuming any global configuration knowledge. This algorithm is flexible enough to deal with changes of configuration, and can resolve conflicts between locally selected actions and manage multiple active hormones for producing coherent global effects.

2 Introduction

Metamorphic or self-reconfigurable robots are highly desirable in tasks such as fire fighting, search and rescue after an earthquake, and battlefield reconnaissance, where robots must encounter unexpected situations and obstacles and perform tasks that are difficult for fixed-shape robots. For example, to maneuver through difficult terrain, a metamorphic robot may transform into a snake to pass through a narrow passage, grow a few legs to climb over an obstacle, or become a ball to roll down a slope. Similarly, to enter a room through a closed door, a self-reconfigurable robot may disassemble itself into a set of smaller units, crawl under the door, and then reassemble itself in the room. To rescue a child trapped deep in rubble in an earthquake, a set of small robots may form a large structure in order to carry an oxygen cylinder that would be too heavy for any individual robot. As an example of such robots, http://www.isi.edu/conro illustrates pictures and movies of the CONRO self-reconfigurable robots.

Metamorphic robots are constructed from a set of autonomous and connectable modules. Although the physical realization of such robots is relatively new, much literature exists for their control in simulation or in robots that have very limited reconfiguration ability. [1-15] are examples of the related works. This paper describes a method for distributed software control of such robots, based on the biological concept of hormones.

This paper extends the research that has been presented in our previous papers [16, 17] in the following aspects. In the algorithms presented in our previous work it was assumed that all modules know the structure of the robot. In this paper we have relaxed this assumption. Modules based on their types are able to locally decide what type of gait they should generate. In addition, our previous approach to synchronization was based on a broadcast/convergecast implementation; the limitation of this approach is that there is only a one-way relationship among modules. For example, when a module sends a gait-specifying hormone to another module, the selected action by the receiver is unknown to the sender and similarly when a synchronization hormone is sent back, the synchronization status of the receiver is unknown to the sender. In this paper we present a new parallel method of synchronization. We also study the task of moving caterpillar with a turn action, in which multiple hormones must be generated and coordinated. This provides a solution to the problem of conflict resolution between multiple active hormones in any decentralized systems.

3 CONRO System Overview

The CONRO self-reconfigurable robots are made of a set of connectable modules. Each module is an autonomous unit that contains two batteries, one STAMP II micro-controller, two motors, four pairs of IR transmitters/receivers for communication and proximity sensing in docking and four docking connectors to allow connections with other modules. Each module has two degrees of freedom: DOF1 for pitch (up and down) and DOF2 for yaw (left and right). Each male connector consists of two pins. More information about the CONRO robot hardware can be found in [17]. Fig.1 shows the mechanical and schematic view of a module.

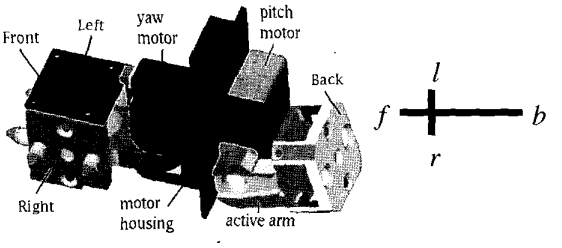


Figure 1: Mechanical and schematic view of a module

Modules can be connected together by their docking connectors, called *links*, located at either end of each module. At one end, called *back* (*b* for short), there is a female connector, consisting of two holes for accepting another module's docking pins. At the other end, three male connectors are located on three sides of the module, called *left* (*l*), *right* (*r*) and *front* (*f*). An *active link* is a

connector that is connected to another module and the connected module is called a *neighbor*. Each active link has a buffer for keeping the received hormones. The name of the buffer is a pair of link names, starting with the module's link name. For example, the name of a module's left link buffer connected to the back link of its neighbor will be *lb*. This name will be used to label the received hormones to that link. Each module has a *type*. The module type is determined based on how active links are connected to the neighboring modules. For example, if a module's back is connected to the left of another module, it will be of type T5. Each module can determine its type locally by checking to which links of the neighboring modules it is connected. This information is communicated among neighbors. There are 32 distinct types for a CONRO module, which have been shown in Table1.

Connected to neighboring modules

b	*f*	*r*	*l*	Type
f				T1
	b			T2
		b		T3
			b	T4
l				T5
r				T6
	b	*b*		T7
		b	*b*	T8
	b		*b*	T9
l	*b*			T10
l		*b*		T11
l			*b*	T12
r	*b*			T13
r		*b*		T14
r			*b*	T15

b	*f*	*r*	*l*	Type
f		*b*		T17
f			*b*	T18
	b	*b*	*b*	T19
f	*b*	*b*		T20
f		*b*	*b*	T21
f	*b*		*b*	T22
l	*b*	*b*		T23
l		*b*	*b*	T24
l	*b*		*b*	T25
r	*b*	*b*		T26
r		*b*	*b*	T27
r	*b*		*b*	T28
f	*b*	*b*	*b*	T29
l	*b*	*b*	*b*	T30
r	*b*	*b*	*b*	T31

Table 1: Module types based on connected links

In parallel with the hardware implementation of the CONRO robot, we have also used a Newtonian mechanics based simulator, Working Model 3D, to develop the hormone-based control theory, with the objective that the theory and its related algorithms will eventually be migrated to the real robots. Working Model 3D is a three-dimensional dynamics simulation program. Using it, a designer can define objects with complex physical properties, including mass, coefficient of friction, moments of inertia, and velocities. Constraints among objects include rigid joints, revolute joints, and linear constraints, including rods, springs, and dampers. User-defined forces, torques, actuators and motors are also available. A more detailed description of the simulator can be found in [18]. Fig2 shows a simulated CONRO caterpillar robot.

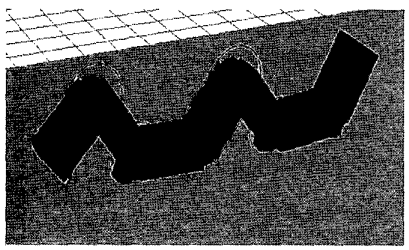


Figure 2: A simulated six-module caterpillar

A self-reconfigurable robot can be viewed as a network of autonomous systems with communication links between modules. The topology of this network is dynamic because a robot may choose to reconfigure itself at any time.

4 Hormone-based Distributed Control

One of the approaches previously used for controlling self-reconfigurable robots is a centralized gait control table [8]. Although it is a comprehensive and simple mechanism, it is not designed to deal with the dynamic nature of robot configuration. Gait controls must be set up in advance, and every time the configuration is changed, no matter how slight the modification is, the control table must be rewritten.

To increase the flexibility of controlling self-reconfigurable robots, we have designed and implemented a new control mechanism based on the biological concept of hormones. Similar to a content-based message [19], a hormone is a message that triggers different actions at different subsystems and yet leaves the execution and coordination of these actions to the local subsystems. For example, when a human experiences sudden fear, a hormone released by the brain causes different actions, e.g., the mouth opens and the legs jump. Formally, a hormone is a type of message that has three important properties: (1) it has no destination but floats in a distributed system; (2) it has a lifetime; and (3) it can trigger different actions at different receiving sites. The actions caused by a hormone may include modification and relay of other hormones, execution of certain local actions, or just ignoring the received hormone. In terms of functions, hormones can be classified by their purposes. Hormone-based distributed control mechanism reduces the communication cost for locomotion controls, yet maintains global synchronization and execution monitoring. In this paper we will not address the effects of the hormone lifetime on the behavior of robot and assume that the lifetime is long enough.

It is worth to point out that hormones used in metamorphic robots are very different from broadcast messages or content-based messages. Hormones are "propagated" signals that may be modified, delayed, or disappeared along the way of propagating from the source to the rest of the entire global system. With this unique property, hormones can be used to accomplish tasks that are beyond the abilities of conventional address-based or content-based messages (regardless if they are broadcast or not), in which all modules receive the same copy of the message.

In [16, 17] we introduced hormone-based gait generation algorithms for caterpillar and spider structures. In those algorithms it was assumed that the gait initiator knows the structure of the robot so that it could generate the appropriate *move* hormone for each robot structure. In this paper we will present algorithms in which there is only one *move* hormone for all types of structures and modules

based on their types are able to decide locally what type of gait they should generate.

In our previous work synchronization was implemented using a broadcast/convergecast effect, in which the gait generator module sends a 'gait hormone' to its neighbors and they, in turn, send gait hormones to their neighbors and wait to receive 'synchronization hormones' back from them. The limitation of this approach is that there is only a one-way relationship among modules. When a module sends a gait hormone to another module, the selected action by the receiver is unknown to the sender so the actions should be pre-specified in such a way that there is no conflict among modules' actions. Similarly, when a synchronization hormone is sent back in response to a gait hormone, the synchronization status of the receiver of the synchronization hormone is unknown to the sender. Here we present a new parallel method of synchronization that is used for choosing the right action and synchronizing its starting time and addresses the above-mentioned issues.

The following subsections explain how hormones propagate in the system and how they are used for task and action specification and synchronization. We will use a four-module caterpillar (such as the one shown in Figure 2) as an example to demonstrate the algorithms.

4.1 Hormone Propagation

Hormone propagation is a mechanism by which a module communicates with other modules. In two situations a module propagates a hormone. The first case is when the module generates a new hormone. In this case, the generator module will send (local broadcast) the hormone to *all* its active links. The second case is when a module receives a hormone and decides to relay it to its other neighbors. In this case, the module marks the receiving link as the *InLink* and will send the hormone to the rest of its active links, called *OutLinks*. Hormone propagation is the only way a module can communicate with other modules. It is not possible for a module to send a hormone to "some" of its active links. It must be either propagated or completely ignored. After a module propagates a hormone, a copy of the hormone remains in the module and its Life-Time (LT) field will be incremented by a local counter in the module and as long as its lifetime is not reached the maximum (maxTime), it remains active. Hormone propagation is used in task specification, synchronization, and conflict resolution.

4.2 Hormones for Synchronization

Synchronization is a general problem for distributed systems. In a master control system [8], synchronization is an operation with a high cost of communication. In a masterless control system [8], it demands an unrealistic assumption that all modules' internal clocks are synchronized.

In a hormone-based control system, solutions to the synchronization problem are naturally suggested by the flexible interpretations of hormones. Since hormones can "wait" at a site for the occurrence of certain events before traveling further, they can be used as tokens for synchronizing events between modules. For example, to synchronize steps in a caterpillar move, a synchronization hormone *s* can be designed to ensure that all modules finish their job before the next step begins.

Two actions can be synchronized in many ways. According to [20] there are thirteen types of temporal relationships among actions. Two of them are 'Starts' and 'Meets', which in this paper are called 'Parallel' and 'Serial' relationships respectively. Parallel actions are those that start at the same time and serial actions are those that one starts after the other ends. These two cases are of our interest since they accomplish almost all synchronization requirements of a metamorphic robot. In the next section we will describe the parallel synchronization algorithm, in which the parallel action-synchronization is used as an example. It can also be applied to other tasks that need parallel synchronization such as gait-selection or self-reconfiguration.

4.2.1 Parallel synchronization

In parallel synchronization many actions need to start at the same time, for example a spider robot should move its legs simultaneously to perform a successful 'walking' gait. Therefore a mechanism is required to signal modules when they should start. After performing the current action, a new action will again be selected and started synchronously.

For this purpose, we have developed a hormone based synchronization algorithm, which runs on each module in parallel and guarantees the same starting time for all synchronized actions. Generally, for a given task, a module can infer that *all* other modules have selected and ready to start their actions when it receives the 'expected' number of action-selection or synchronization hormones from all of its neighbors. The expected number of hormones for a neighbor is the number of active links of that neighbor.

Two types of hormones are important for synchronization purpose, they are action-specifying hormones (ASH) and synchronization hormones (SH), as illustrated in Figure 3.

ASH(task, action, LT, maxTime)

SH(task, label (list of hormones), LT, maxTime)

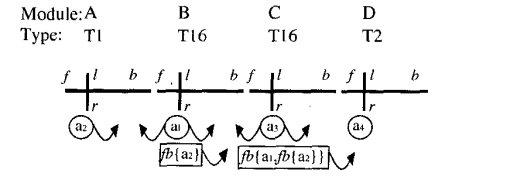
Figure 3: Action-Specifying and Synchronization Hormone Format

In performing a task, when a module selects an action, it generates and propagates an action-specifying hormone, e.g. **ASH**(move, a_1 ,0, 10). Each link stores the received hormones from its neighbor in its buffer. When the number of stored hormones is equal to the number of the neighbor's active links (i.e., the expected number of

hormones), the module sends a synchronization hormone to the rest of the neighbors. The new hormone contains the received hormones labeled with the buffer name. For example, if the above action-specifying hormone is received from a neighbor with one active link and the buffer name is *lb*, the synchronization hormone will be:

SH(move, *lb*(**ASH**(move, a_1 ,0, 10)), 0, 10)

Receivers will count the synchronization hormone as one received hormone. Although the content of the ASH and SH are not required for the synchronization purpose, they will be kept by modules and will be used in conflict resolution, which will be described in the later sections. The following is a detailed example of parallel action synchronization of a four-module caterpillar robot, Figure 4. For simplicity, it is assumed that the robot is performing only one task, and only actions are represented in the internal representation of received hormones. The type of each module is shown according to Table 1. The number of active links of modules A to D is 1-2-2-1, respectively, which defines the "expected" number of hormones for each connection. For example, A expects 2 hormones from its *bf* neighbor module B (for B has 2 active links). C expects 2 hormones from its *fb* neighbor (B) and 1 hormone from its *bf* neighbor D (for D has 1 active link). Figure 4a shows the hormones that have been propagated. **ASH**s are shown in circles and **SH**s are shown in rectangles. Notice that module 'B' ('C') has received the expected number of hormones from 'A' ('B') and has propagated a **SH** to 'C' ('D'). The lower part of the figure 4a shows the buffer contents of the modules after receiving the hormones. The '?' sign represents hormones that are expected. For example, B has two active link buffers: *fb* expected 1 and received 1 hormone, a_2, and *bf* expected 2 but received 1 hormone, a_3 (so it has a "?"). This synchronization process is complete when 'D' propagates its action, a_4. It causes modules 'C', 'B', and 'A' to receive all of the expected hormones and together with module 'D', start execution of their actions. Figure 4b shows the internal representation of modules after completion of the synchronization process.



(a)

Module	*Internal representation*
A →	[a_2, [***bf***{a_1, *bf*{a_3, *bf*{a_4}}}]]
B →	[a_1, [***fb***{a_2}, ***bf***{a_3, *bf*{a_4}}]]
C →	[a_3, [***bf***{a_4}, ***fb***{a_1, *fb*{a_2}}]]
D →	[a_4, [***fb***{a_3, *fb*{a_1, *fb*{a_2}}}]]

(b)

Figure 4: An example of parallel synchronization

Compared to the centralized control system with a standard message passing protocol, which requires $O(n^2)$ message hops for each synchronized action (because n messages must be sent to n modules), hormone-based mechanism requires only $O(t^2n)$ hops (Because each module generates t messages and relays $t(t-1)$ messages), where n is the number of modules in the configuration and t is the number of its active links, where $1 <= t <= 4$. The total time for synchronization is $O(cn+L)$, where c is the constant processing time for sending out a message, and L is the longest time needed for a module to complete its local action in the current configuration.

4.2.2 Serial synchronization

When serial synchronization among the actions of a task is required, a synchronization hormone will be sent after the action is done and other modules will start their actions on receiving the hormone.

4.3 Gait Representation

A gait is defined as one cycle of a pattern of motion that is used to achieve locomotion [8]. In a self-reconfigurable robot, which can be viewed as a distributed network of autonomous modules, motion is generated when modules perform actions. Desired pattern is created when the right actions are performed at the right time. Therefore the definition of a gait should be in such a way that enables the robot's modules to select the right action respect to other modules and create a cyclic pattern of actions. Based on these requirements a gait should contain the following information:

- Synchronization information.
- A set of actions and an ordering among them.
- A set of constraints on the modules selected actions respect to other modules.

Setting the angle of a DOF to a predefined value is the action that a module can perform. It is possible to define functions like *next()*, or *previous()* on actions, which returns the next or previous action according to the actions order. A constraint has the following form: (<action> <relation> <link-pairs sequence> <action>). The role of constraints is to create pattern of actions among modules by eliminating the undesirable action candidates.

For example, a caterpillar move gait is defined as:

- Synchronization: Parallel.
- Actions: An ordered circular list of actions (a_0-a_3), which sets the DOF1 to the specified value: ($a_0 \rightarrow 0°$, $a_1 \rightarrow 30°$, $a_2 \rightarrow -60°$, $a_3 \rightarrow 30°$)
- Constraint: (a_x= *fb next*(a_x)). This constraint means that if the *f* link of a module, *i*, is connected to the *b* link of another module, *j*, then the action of a module *j* should be the next action of module *i*.

There are two symbols that can be used instead of the link-pairs in a constraint. The first one is the '+' symbol

which means one or more link-pairs and the second is '*' symbol which means zero or more link-pairs.

4.4 Hormones for Task Specification

The unique properties of hormones make them ideal for specifying tasks in a distributed system with near-minimal communications. This process is initialized by a task-specifying hormone **TSH**, whose format is illustrated below:

TSH(task, gait, module-type, LT, maxTime)

Upon receiving a task-specifying hormone, a module will select a gait based on its type. The type of a module usually gives useful information about the type of gait that a module can perform. For example, if all modules of a robot are of type T1, T2, and T16, which creates a caterpillar shape robot, then it will be able to perform a caterpillar gait. In situations where a module can perform more than one gait, the module will select what is more suitable for most of the modules. Therefore if gait selection is done synchronously all modules can find out what other modules have selected and choose the right gait. For example, when module A in Figure 4a receives **TSH**(move, null, null, 0, 10), it will select a caterpillar gait, and generate/propagate a new task-specifying hormone, **TSH**(move, caterpillar, T1, 0, 10) to other modules. When all modules complete the parallel synchronization, they have selected the right gait and are ready to select their actions. Action-selection will also go through a synchronization process, and when all modules complete their action selection, they start executing their actions. This loop of action selection, synchronization, and execution will continue until new task-specification hormones are received.

4.5 Conflict Resolution

In the above algorithm it was assumed that when modules select and synchronize their actions, there is no conflict among the selected actions. In reality, since modules select actions independently, it is possible that actions of two modules violate some constraints in their gait definition. Therefore a conflict resolution phase is required.

The first step in conflict resolution is the *constraint checking*. It consists of checking the selected actions in the internal representation, gathered during the action selection phase such as the ones shown in Figure 4, against the constraints. A constraint matches the internal representation of a module if there is an exact match between the link-pairs of a constraint and a sequence of labels connecting two actions in the internal representation. For example, the constraint given in the caterpillar gait, i.e. (a_x) = *fb* $next(a_x)$ matches "a_1 *fb* a_2" because $a_x = a_1$ and $next(a_x) = a_2$ according to the action order specified in the gait. However, "a_3 *fb* a_1" does not match the constraint, and one of the actions must be changed.

Decision about which action needs to be changed is based on the Life-Time (LT) of the action-specifying hormone. Between the two actions the one with smaller LT value will be selected. If conflict is detected before propagation, the selected action will be changed and a consistent action will be propagated. However, in situations that the action is already propagated, a conflict-resolution hormone (**CRH**) will be propagated. The format of **CRH** is shown as follows:

CRH(task, **ASH**, constraint , LT, maxTime).

A **CRH** contains the conflicting hormone and the violated constraint. In the above example, assuming that the hormone containing a3 has a smaller LT, the generated **CRH** is:

CRH(move, **ASH**(move, a_3, 0, 10), (a_3 *fb* a_1), 0, 10)

If a receiving module has the conflicting **ASH** in its internal representation, it will delete that hormone, update the number of received hormones in the receiver buffer, and propagate the **CRH**. Otherwise, the receiving module will ignore the received **CRH**.

When the module whose action is the source of conflict receives the **CRH**, it will select an action that satisfies the constraint included in the **CRH** and generates a new **ASH** containing the new action and propagate the hormone. In the above example, module 'C', which selected a_3, will re-select action a_0 and generate a new **ASH**. If module 'D' receives the new **ASH** before propagating its selected action, it will select a consistent action, which in this case is a_4.

4.6 The CELL Algorithm

All activities discussed above are parts of the CELL algorithm described below, Figure 5. This algorithm runs locally and autonomously at each module.

```
When a (task) received do
{
    Select a gait with Parallel Synchronization & Conflict Resolution;
    If (the synchronization of the selected gait is "Parallel")
        {
            selectAction(selectedGait)
            While (task-specification hormone is active)
             {
                    propagate(selectedAction);
                    parallelSynchronization&ConflictResolution();
                    execute(selectedAction);
                    selectNextAction(selectedAction, selectedGait);
              }
        }
    If (the synchronization of the selected gait is "Serial")
        {
            selectAction(selectedGait);
            While (task-specification hormone is active)
            {
                execute(selectedAction);
                propagate(selectedAction);
                selectNextAction(selectedAction, selectedGait)
            }
        }
}
```

Figure 5: Hormone-based Control Algorithm, CELL.

4.7 Multiple Hormone Management

In [16] we presented hormone-based algorithms for spider and caterpillar gait generation with a single active hormone. To support multiple hormone generation and management, we illustrate briefly in this section how CELL algorithm uses two synchronized hormones to perform both moving and turning in a caterpillar configuration. The "caterpillar-move" gait described above will move the robot along its body and the "caterpillar-turn" gait, as described below, will bend the body to the side. The synchronized combination of these two gaits will generate a circular trajectory. The "caterpillar-turn" gait is defined as:

- Synchronization: Parallel.
- Actions: A list of actions (a_5-a_6), which sets the DOF2 to the specified value: ($a_5 \rightarrow 0°$, $a_6 \rightarrow 10°$)
- Constraint: (a_x) != $+a_6$, which means there cannot be two modules in the robot that perform the a_6 action at the same time.
- Local constraints: Synchronization between two gaits are performed using the following local constraints:
  if (a_x=a_4 & a_y = a_6) then $next(a_y) = bf(a_y)$
  if (a_x=a_3) then $next(a_y) = bf(a_y)$

The last two constraints are used to identify the next action of a module. They ensure that the next action of a module is the action performed by its back neighbor. Along with the constraint that restricts the bending to one and only one module, this gait will shift the bent-action between a unique pair of modules who are performing the caterpillar move.

5 Conclusion and future work

In this paper, we have described a hormone-based, distributed algorithm for controlling the selection, synchronization, and execution of actions in a metamorphic robot. This algorithm has been implemented and demonstrated in the 3D Model simulation environment for locomotion in various configurations (e.g., snakes with different length and insects with different numbers of legs). Our future work includes the application of CELL algorithm to self-reconfiguration tasks such as shape changing from a legged configuration to a snake, and vice versa.

6 Acknowledgements

This research is sponsored in part by DARPA/MTO under contract number DAAN02-98-C-4032, and in part by AFOSR under award number F49620-97-1-0501.

7 References

1. Fujita, M., H. Kitano and K. Kageyama. *Reconifurable physical agents.* in *Autonomous Agents.* 1998.
2. Fukuda, T., and Y. Kawauchi. *Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator.* in *ICRA.* 1990.
3. Kotay, K., D. Rus, M. Vona, and C. McGray. *The self-reconfiguring robotic molecule.* in *ICRA.* 1998.
4. Murata, S., H. Kurokawa, E. Toshida, K. Tomita, and S. Kokaji. *A 3-D self-reconfigurable structure.* in *ICRA.* 1998.
5. Neville, B., and A. Anderson. *Tetrabot family tree: Modular synthesis of kinematic structures for parallel robotics.* in *Proceedings of the IEEE International Symposium on Robotics Research.* 1996.
6. Pamecha, A., I. Ebert-Uphoff, G.S. Chirikjian, *Useful Metrics for Modular Robot Motion Planning.* IEEE Trans. on Robotics and Automation, 1997. **13**(4): p. 531-545.
7. Paredis, C., and P. Khosla. *Design of modular fault tolerant manipulators.* in *Proceedings of the First Workshop on Algorithmic Foundations of Robotics.* 1995.
8. Yim, M., *Locomotion with a unit-modular reconfigurable robot (Ph.D. Thesis),* in *Department of Mechanical Engineering.* 1994, Stanford University.
9. Yoshida, E., S. Murata, K. Tomita, H. Kurokawa, and S. Kokaji. *Distributed formation control of a modular mechanical system.* in *IC-IROS.* 1997.
10. Bojinov, H., A.Casal,T.Hogg. *Multiagent Control of Self-reconfigurable Robots.* in *ICMAS.* 2000. Boston,MA,USA.
11. Lee, W.H., Sanderson A. C. *Dynamic Rolling, Locomotion Planning, and Control of an Icosahedral Modular Robot.* in *International Conference on Intelligent Robots and Systems.* 2000.
12. Castano, A., Will P. *Mechanical Design of a Module for Reconfigurable Robots.* in *IC-IROS.* 2000.
13. Kotay, K., Rus D. *Algorithms for Self-reconfiguring Molecule Motion Planning.* in *IC-IROS.* 2000.
14. Rus, D., Vona M.,. *A Basis for Self-Reconfiguring Robots Using Crystal Modules.* in *IC-IROS.* 2000.
15. Satoshi, M., et al. *Hardware Design of Modular Robotic System.* in *IC-IROS.* 2000.
16. Shen, W.-M., B. Salemi, and P. Will. *Hormones for Self-Reconfigurable Robots.* in *Intelligent Autonomous Systems.* 2000. Venice, Italy.
17. Shen, W.-M., Y. Lu, and P. Will. *Hormone-Based Control for Self-Reconfigurable Robots.* in *Autonomous Agents.* 2000. Barcelona, Spain.
18. *Knowledge Revolution Working Model 3D User's Manual Version 3.0,* . 1997.
19. Hirokazu, I., and K. Mori, *Autonomous Decentralized Computer Control Systems.* IEEE Computer, 1984.
20. Allen, J.F., *Time and Time Again: The Many Ways to Represent Time.* Intl J. of Intelligent Systems, 1991.