Hormone-Based Control for Self-Reconfigurable Robots

Wei-Min Shen, Yimin Lu, Peter Will
Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292, USA
310-822-1511
{shen,lym,will}@isi.edu

ABSTRACT

Self-reconfigurable or metamorphic robots can change their individual and collective shape and size to meet operational demands. Since these robots are constructed from a set of autonomous and connectable modules (or agents), controlling them is a challenging task. The difficulties stem from the facts that all locomotion, perception, and decision making must be distributed among a network of modules, that this network has a dynamic topology, that each individual module has only limited resources, and that the coordination between modules is highly complex and diverse. To meet these challenges, this paper presents a distributed control mechanism inspired by the concept of hormones in biological systems. We view hormones as special messages that can trigger different actions in different modules, and we exploit such properties to coordinate motions and perform reconfiguration in the context of limited communications and dynamic network topologies. The paper develops a primitive theory of hormone-based control, reports the experimental results of applying such a control mechanism to our CONRO metamorphic robots, and discusses the generality of the approach for a larger class of distributed autonomous systems.

Keywords

Self-reconfigurable robot, multi-agent control, collaboration, gaits, locomotion, reconfiguration, robotics, autonomous agents.

1. INTRODUCTION

Self-reconfigurable robots can change their individual and collective shape and size to meet operational demands. Such metamorphic robots are highly desirable in tasks such as fire fighting, search and rescue after an earthquake, and battlefield reconnaissance, where robots must go through 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 to carry cooperatively an oxygen cylinder that would be too heavy for any individual robot. As an example of such robots, Figure 1 shows a CONRO robot, in a configuration of 9 modules, developed at USC/ISI.

Figure 1: A CONRO Prototype Robot

Self-reconfigurable 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. The simplest control of self-reconfigurable robots is to tether all the modules to a centralized controller. Several authors have adopted this simplified approach to address the feasibility of designing robot molecules [1] or the metric properties of reconfigurable space [2]. Yim [3, 4] designs a control mechanism using a gait control table to specify each DOF action at a given step. These tables are simple to construct and easy to use, but they must be rewritten if the configuration is modified. Using these tables, Yim also proposes a masterless control mechanism assuming that all modules' internal clocks are synchronized and that no other unexpected events occur during the execution of a table. This approach is free from intra-module communication, but there is a loss of robustness since all modules run "open-loop" and there is no room for high-level control actions such as topology discovery or determining optimal configurations for a given task. Fukuda and Kawauchi [5] propose a cellular robotics system to coordinate a set of specialized modules. Fujita, et. al. [6] describe a reconfigurable robot dog. However, the ability of autonomous reconfiguration is not emphasized in the paper. Fahlman [7] addresses the control of distributed systems in the context of knowledge representation. In his NETL system, he uses markers to dynamically form sets for performing parallel operations. Arkin [8] uses hormones in robot control, but this paper extends his research with several innovations both in algorithms and applications to self-reconfigurable robots.

Much research exists for the design of metamorphic robots. Fukuda and Kawauchi [5] proposed a cellular robotic system to coordinate a set of specialized modules. Murata, et al. [9] and Yoshida, et al. [10], designed and constructed separate systems that can achieve planar motion by arranging modules. Paredis and Khosla [11] proposed modular components for building fault-tolerant multipurpose robots. Neville and Sanderson [12] proposed a module for the dynamic construction of complex structures. As more and more self-reconfigurable robots are emerging from the research, the need for a simple yet general control mechanism for this type of distributed system becomes increasingly urgent.

Our research has focused on both the hardware and software of self-reconfigurable robots. This paper, however, will emphasize the software control of such robots. In a distributed environment where system configuration is dynamic and resources of individual modules are very limited, the control software must provide support for the control of locomotion, docking, morphing, and fault tolerance. To meet these challenges, this paper presents a distributed control mechanism inspired by the concept of hormone in biological systems. We view hormones as special messages that can trigger different actions in different modules, and we exploit such properties to coordinate motions and reconfiguration in the context of limited communications and dynamic network topologies. The paper develops a primitive theory of hormone-based control, reports the experimental results of applying such a control mechanism to our CONRO metamorphic robots, and discusses the generality of the approach for a larger class of distributed autonomous systems.

The rest of the paper is organized as follows. Section 2 gives an overview of our CONRO robots. Sections 3 and 4 discuss the communication and control issues in the self-reconfigurable robots in general. Section 5 introduces the hormone-based control, and describes in detail its application to action specification, synchronization, and dynamic grouping. Section 6 deals with the issues in hormone management. Section 7 discusses the control of reconfiguration. Section 8 describes our current implementation and reports experimental results on the CONRO robots. The paper concludes with a short discussion of our future research directions, and the possible application of hormones to distributed autonomous systems in general.

2. OVERVIEW OF THE CONRO ROBOT

This section gives a brief overview of the CONRO robot. For more information about CONRO, readers are referred to [13],[14] and our web site: http://www.isi.edu/conro. The CONRO self-reconfigurable robots are made of a set of connectable modules. As illustrated in Figure 2, each module is an autonomous unit containing two batteries, one micro-processor, two motors, four pairs of IR transmitters/receivers, and four docking connectors to allow connecting with other modules.

Modules can be connected together by their docking connectors, located at either end of each module. At one end, three male connectors are located on three sides of the module. Each male connector consists of two pins (see the solid triangles in Figure 3). At the other end, a female connector is located at the tip of the module. It consists of two holes (see the hollow triangles in Figure 3) for accepting another module's docking pins. The female connector has a locking/releasing mechanism behind the holes, and can have two states. In the default or non-active state, it can accept and lock the incoming pins by a spring motion. In the activated state, it can release the lock by triggering an SMA actuator. To illustrate the connections between modules, Figure 3 shows the possible connections from a top-down point of view.

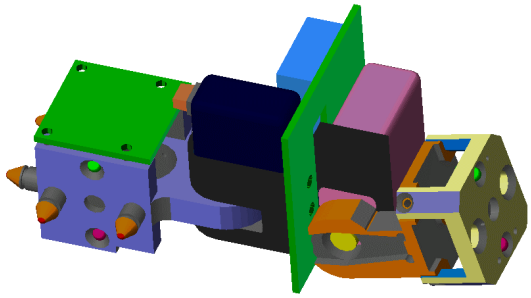

Figure 2: A CONRO module

Each module has two degrees of freedom: DOF1 for pitch (up and down) and DOF2 for yaw (left and right). Each DOF has a home position (when the joint is straight), and has two joint limits (when the joint reaches the maximal or the minimal angle). With these two DOFs, a single module can wiggle its body but cannot change its location. However, when two or more modules connect to form a structure, they can accomplish many different types of locomotion. For example, a body of six legs (see Figure 1) can perform hexapod gaits, while a chain of modules can mimic a snake or a caterpillar motion. Figure 4 illustrates a 6-module caterpillar (or "nessie") motion. To move the combined modules forward, each module's DOF1 goes through a series of positions, and the synchronized global effect of these local motions is a forward movement of the whole caterpillar (indicated by the arrow).

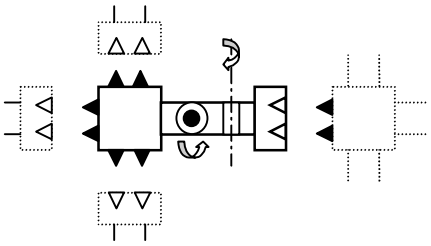

Figure 3: Possible docking connections between modules

To completely specify this gait, one can use a "gait control table" as suggested in [3], where each row in the table corresponds to the target position for all DOFs in the configuration during a step. Each column corresponds to the sequence of desired positions for one DOF. The control starts out at the first step in the table, and then switches to the next step when all DOFs have reached their target position in the current step. When the last step in the table is done, the control starts over again at step 0. Table 1 lists the control for the caterpillar movement in Figure 4. As we can see in this table, the six columns correspond to the six modules' DOF1 (the leftmost is M1, and the rightmost is M6). The first row in this table corresponds to Step 0 in Figure 4.

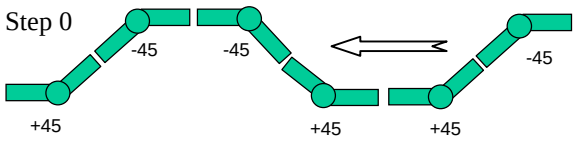

Figure 4: A caterpillar (or nessie) movement

The CONRO modules communicates with one another using IR transmitters and receivers. Each connector contains an IR transmitter/receiver pair (shown as a red hole and a green hole in Figure 2) located between the pins/holes. When a module is connected to another module via the connectors, the two pairs of IR transmitters/receivers at the docked connectors will be aligned to form a bi-directional communication link. Since there are four connectors for each module, there can be up to four communication links for each module.

Table 1: The gait control table for the caterpillar movement

Step	Module ID for DOF1					
	M1	M2	M3	M4	M5	M6
0	+45°	-45°	-45°	+45°	+45°	-45°
1	-45°	-45°	+45°	+45°	-45°	-45°
2	-45°	+45°	+45°	-45°	-45°	+45°
3	+45°	+45°	-45°	-45°	+45°	+45°

The IR transmitters/receivers can also be used as docking approximator sensors for guiding module connection during a reconfiguration action. For example, when a chain of modules (a snake) wants to become a loop of modules (a rolling track), the tail module must seek for and then dock to the head module. To do so, the body modules will first bend, to bring the head and the tail modules into the same approximate vicinity. Then the head module's IR transmitter will transmit IR signals (much like a TV remote controller) and the tail module will use its IR receivers (by calling a local routine IR_Approximate_Sense(M_{tail})) to seek the source of the signal in order to permit docking. The strength of the received signal indicates the closeness of the transmitter. Note that in a self-reconfigurable robot, the control of reconfiguration and the control of locomotion are the same task from a module's point of view [2]. Thus, one can imagine using a gait control table to control the docking procedure described above. A more difficult reconfiguration task is to have a snake "grow" a leg. In this task, the tail module must be docked to a body module (forming a T connection), and then a module near the tail (depending on how long the leg needs to be) must disconnect from its neighbor module to become the end module of the leg.

3. COMMUNICATION ISSUES

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 reconfigure itself any time it chooses to. The use of routing tables is not feasible in this network at present, because our current individual modules have very limited computational resources.

To enable communication in this network, we assume that each module has a unique ID and maintains a set of active communication links (we say a link is active if it is linked to another module). Since each link corresponds to a physical connection, we define the configuration of a self-reconfigurable robot to be the configuration of this network. In theory, this corresponds to a graph in which the nodes are labeled modules and the edges are active links.

To send a message from a module P to a module Q in this network, P sends the message (with the destination set to Q) to all of its active links. Upon receiving a message, a module will either keep the message if the message's destination matches its own ID, or relay the message to all of its active links except the link through which the message was received. (This link is called the *inlink* of the message; all other active links of the module are called *outlinks* of the message.) Broadcast messages are assigned a special destination ID so that they will be processed and relayed by the modules. In a network that contains loops, messages may circulate indefinitely, and such a case must be prevented by the control mechanism.

Since communication among modules is a critical factor for a self-reconfigurable robot, we measure the *communication cost* of a message by the number of hops before it reaches its destination. Notice that this includes all the hops, regardless of whether they are on the correct paths or not. With this definition, we can compare the communication costs of two different control mechanisms by measuring the total communication costs each requires to accomplish the same task.

4. MASTER VS. MASTERLESS CONTROL

Given a gait control table, previous researchers have proposed both master control and masterless control [3]. In a master controlled system, a designated control unit sends commands to individual modules and synchronizes their actions. For example, to move from Step 0 (see Figure 3) to Step 1 in Table 1, the master sends six messages to the six modules in the configuration, and then waits for six replies from the modules to ensure that all actions have been completed so that the next step can start. The advantage of this approach is that all modules are synchronized and the master can monitor the task progress to regain some fault tolerance. The downside, however, is the cost of communications. In this particular example, each step requires n messages (where n is the total number of modules in the configuration). Since each message needs $O(n)$ hops, the communication cost of a single step in this configuration is $O(n^2)$, for there are n messages to be sent and each message needs $O(n)$ hops. Another disadvantage of this mechanism is the dependence upon communication. In a distributed environment such as self-reconfigurable robots, where communication is not always reliable and timely, this type of control may not be able to deliver its full potential in reality.

In a masterless control system, each module manages its own actions. In a gait control table, this means that a module can execute its own column without waiting for external commands. To ensure synchronization between modules, one must assume that every module has an internal clock that is synchronized with the others, and that each step in the control table has a duration that is agreed across all modules in the configuration. The advantage of this totally distributed approach is that it is free from communication and thus scalable to large systems that have many modules. Its cost, however, is a loss of robustness since all

modules run "open-loop" and there is no measurement of task progress. Furthermore, since all modules in a self-reconfigurable robot are autonomous, it is very difficult, if not impossible, to ensure that all modules' internal clocks are synchronized.

5. HORMONE-BASED CONTROL

Although the gait control table is a comprehensive and simple mechanism for controlling self-reconfigurable robots, it is not designed to deal with the dynamic nature of robot configuration. Every time the configuration is changed, no matter how slight the modification, the control table must be rewritten. For example, if two snakes join together to become one, a new control table must be designed from scratch. A simple concatenation of the existing tables may not be appropriate because their steps may mismatch. Furthermore, when robots are moving along rough ground, their actions on each DOF cannot be determined at the outset.

To overcome some limitations of both master and masterless control, we have designed and implemented a new control mechanism for self-reconfigurable robots based on the biological concept of hormone. The basic idea is that a hormone is a signal that triggers different actions in different body parts (modules) and yet leaves the execution and coordination of these actions to 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. Using this property, we designed a control mechanism that lies somewhere between master and masterless control. It reduces the communication cost for locomotion controls, yet maintains some degree of global synchronization and execution monitoring.

Formally, a hormone message is a special type of message that has the three important properties: (1) a hormone has no particular destination but floats in a distributed system; (2) a hormone has a lifetime; and (3) the same hormone may trigger different actions at different receiving sites. These actions may include modification and relay of the hormone, execution of certain local actions, or destruction of the received hormone. A hormone is terminated in three possible ways: when it reaches its destination, when its lifetime expires, or when it has nowhere to go (e.g., it arrives at a module that has no outlinks). Since no hormone can live forever, this prevents them from circulating in the network indefinitely.

Based on these definitions, we have classified hormone messages into three classes based on their purpose: for action specification, synchronization, and dynamic grouping of modules. We now discuss each of them in detail.

5.1 Hormones for Action Specification

The unique properties of hormones make them ideal for controlling actions in a distributed system such as a self-reconfigurable robot. To illustrate the idea, let us consider the caterpillar example again. This time, we notice that in Table 1 there is a "shifting" pattern among the actions performed by the modules. The action performed by a module M_i at step t is the action to be performed by the module M_{i-1} at step (t+1).

Thus, instead of maintaining the entire control table, we design a hormone message *h(x)* that will cause a receiving module to relay its current DOF1 position to the next module and then move its DOF1 to the *x* position. In this case, the message sent to the next module is a modified hormone *h(y)* where *y* is the DOF1 before the local action is executed. This procedure is illustrated in Figure 5. The tail (rightmost) module M6 acts as the hormone generator (or a temporary local master) and sends a sequence of *h* messages, with different *x* values specified in the M6 column in Table 1.

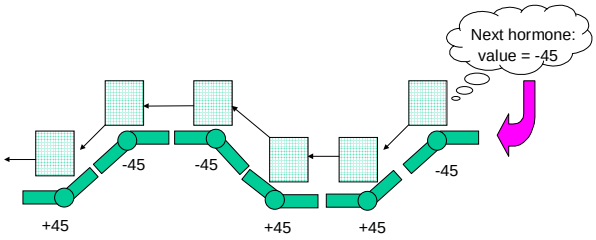

Figure 5: A hormone-controlled caterpillar move

Notice that each *h* message will travel from the tail to the head, regardless of how many modules are in the current configuration. This is an advantage over the gait control tables, for the table for an n-module snake is different from the table for an m-module snake (n≠m), and a new table must be written every time a module is added to or deleted from the snake.

An *h* message is terminated at the head, because that is the only module with no outlinks to relay the hormone. Compared to the master control system with a standard message-passing system, the communication cost of a step here is $O(n)$ because only one *h* message is needed for a step and each *h* message costs *n* hops.

5.2 Hormones for Synchronization

Synchronization is a general problem for distributed systems. In a master control system, it is an operation with a high communication cost. In a masterless control system, 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 travelling 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. When a module receives an *s* hormone, it checks if its local actions are completed or not. If yes, it will relay the *s* hormone to the next neighbor; otherwise, it will wait for the completion of its local actions before it relay the message. In the above example, if such an *s* hormone has traveled from the head to the tail, then the tail module can be sure that all actions in this step have been completed and the next *h* hormone is ready to be sent. Furthermore, the generation of the *s* hormone by the head module can be triggered by the arrival of a new *h* hormone so that the synchronization and actions are coherently controlled by a single mechanism. The complete algorithm for a synchronized caterpillar move is summarized in Algorithm 1.

Loop:

1. Receive a hormone H and remember its inlink and outlink;
2. If H is an *h(x)* hormone,
3. send *h(y)* to the outlink, where *y* is the current DOF1,
4. start moving DOF1 to the target value *x*,
5. If self=head, send self an *s* hormone;
6. If H is an *s* hormone,
7. If DOF1 has not reached the *x* position yet, then wait,
8. If self=tail, then send self the next *h(x)* hormone,
9. else send the *s* hormone to the outlink.

Algorithm 1: Hormone-based control and synchronization

Notice that this algorithm is homogeneous for all modules. This is an important factor in limiting the complexity of a distributed system. If every module needed to run a different control algorithm, then the whole robot would be too complex to control and debug. Notice also that the total time for an *h(x)* to reach from the tail to the head is *O(cn)*, where *n* is the number of modules in the configuration and *c* is a constant processing time for sending out a message. The total time for an *s* hormone to reach from the head to the tail is *O(cn+L)*, where *L* is the longest time for a module to complete its local action in the current configuration.

The above synchronization method can be applied to any Completely Connected Set (CCS) of modules with respect to a synchronization source connector. By CCS, we mean that (1) a module that is connected to the synchronization source X is in the CCS of X, and (2) all modules that are connected to an element in the CCS are also in the CCS. In the example depicted in Figure 4 and 5, the set {M2, M3, M4, M5, M6} is a CCS with respect to the right connector of the head module M1. (Note that M1 itself is not in the CCS.) Similarly, an arm of linked modules forms a CCS with respect to the "shoulder" connector of the arm.

Unlike a CCS, a non-CCS set may include elements that are scattered across the entire body. For example, the set {M1, M3, M5} in Figure 4 is such a set because it does not satisfy the definition of a CCS. Another example of such a set is the set of "three spider legs that have odd number IDs." To synchronize such a set, we must develop mechanisms to group the relevant elements with some "markers" before they can be synchronized using hormones.

5.3 Hormones for Dynamic Grouping

In any distributed system, it is often useful to define a set of entities dynamically as a special group for certain operations. Since the network topology in a self-reconfigurable robot is not static, one cannot foresee all possible ways of grouping modules, but must define them as needed. For example, to synchronize actions in a snake configuration, it is necessary to treat all of the modules in the body as a single set. But in a hexapod configuration, such as the one shown in Figure 1, synchronization of locomotion demands that the six legs be divided into two separate sets.

Inspired by the earlier research in marker-passing systems [7], we designed hormones in such a way that they can be used to define sets on the fly. To do so, we assume that each module in the self-reconfigurable robot has a set of local variables $\{m_i\}$ that can be "marked" dynamically by the module itself. Since hormones are interpreted by receivers based on their own characteristics, hormones can be sent out to cause certain types of module to mark themselves distinctly. For example, to make M1, M3, and M5 form a set in the snake configuration in Figure 3, a hormone can be sent to say "if your ID is an odd number, then mark your m_1 variable to be 1." After that, these modules will react to any future hormone that is meant for "modules with m_1=1." Similarly, to divide the six legs in Figure 1 into two groups, two hormones can be sent out as follows:

If you are a leg module[1] and have an odd ID, then set m_2=1.

If you are a leg module and have an even ID, then set m_3=1.

With this dynamic grouping mechanism, a self-reconfigurable robot can group an arbitrary set of modules into a new group as long as a hormone can be designed to define the characteristics of the relevant modules. The vocabulary for designing new hormones is determined by the scope of the interpreter at each module. We will have more to say about this in the next section on hormone management.

6. HORMONE MANAGEMENT

We have shown that hormones can provide a flexible mechanism for various control problems in self-reconfigurable robots. We now discuss the mechanism at a higher level, namely, how hormones are generated and managed.

We view hormone generation as yet another local module action to be triggered by an incoming hormone or by external sensor stimuli. Since hormones are typically generated as a sequence, once a module becomes a hormone generator, it must know the next hormone in the sequence. In the caterpillar move example (see Figure 5 and Algorithm 1), the tail module is the generator of *h(x)* hormones and it must know how to generate the next *h(x)* hormone at Line 8 of Algorithm 1.

In general, when a module becomes the generator of a particular hormone H, we assume it has the sequence **H** of H, along with an index variable H_i, in its local memory. To generate the next hormone in the sequence, the module simply increments H_i=*mod*(H_i+1, |**H**|), where |**H**| is the length of **H**, and then retrieves **H**[H_i]. To illustrate the idea, consider tail module M6 (see Figure 5), which is the generator of the *h* hormone for the caterpillar move. In this case, M6 has a sequence **H**=[*h*(-45),*h*(-45),*h*(45),*h*(45)]. Assume currently H_i=3, then the next *h* hormone to be generated is **H**[*mod*(3+1,4)]=**H**[0]=*h*(-45), and H_i=0 after the retrieval action.

To ensure the homogeneity of all modules (i.e., any module can become the generator for any hormone sequence), we assume that every module knows all hormones that are defined. To restrict the complexity, however, we assume that no module can be the generator of two or more hormone sequences simultaneously. To implement this, we assume that each module has a local variable *G* to indicate which hormone sequence it is currently generating. A module stops generating hormones if *G* is set to nil. (With this implementation, the condition "self=tail" at Line 5 in Algorithm 1 should be replaced by "*G=h*", and "self=head" at Line 8 by "*G=s*".)

[1] A "leg module" can be characterized as a module that has only one neighbor, and this neighbor has at least three active connections.

A module can become the generator of a hormone sequence in two ways. It can be either self-promoted or instructed. In the self-promoted case, a local sensor routine might be triggered by an external stimulus and assign a value to *G* in response to the trigger. In the instructed case, a module assigns a value to *G* because it is instructed to do so by a special hormone trigger message. Similarly, a module stops producing hormones if G is set to nil, either by self-promotion (because of an external stimulus) or by a special hormone-stopping message. At present, only the instructed way of triggering a module to be a hormone generator is implemented.

The above approach for hormone management offers great flexibility in hormone control. Depending on which module is the hormone generator, the same hormone can produce a wide variety of behaviors. For example, in the caterpillar movement case, if the tail module is the generator of *h* hormones, the caterpillar moves forward. If the head is the generator, the caterpillar moves backward. If a module X in the middle of the body is the generator, then the two segments of the caterpillar (divided by module X) will move apart, in opposite directions. The last action is useful when the caterpillar must disconnect itself at X to become two smaller caterpillars.

Although one module cannot simultaneously generate multiple sequences of hormones, multiple hormones can be simultaneously active in a distributed robot system because they can be generated by different modules. In Algorithm 1, we see that modules are dealing with *h* and *s* hormones simultaneously. Another example of this sort is a forward/turning motion. In this case, one hormone coordinates the forward locomotion while another hormone manages the turning actions of modules.

As the final comment on hormone management, we would like to mention that the number of possible hormones that can be defined in a robot is bounded only by the number of local features that can be sensed by modules. The more sensors a module has, the more types of hormones it can interpret. Of course, all these factorsare also bounded by the local computational resources.

7. RECONFIGURATION CONTROL

We have introduced the concept of hormone control in terms of locomotion planning and execution, without mentioning the actions of reconfiguration. This is because reconfiguration of a self-reconfigurable robot involves the same type of control as that needed for locomotion [2].

The only action that is unique to reconfiguration is the docking (and disconnecting) of two modules. It is interesting from a hormone-control point of view, because whenever there is a choice of control protocols, we would like to select those that are suitable for hormone control. In other words, hormone-based control systems prefer actions across modules to be similar or related in a regular way.

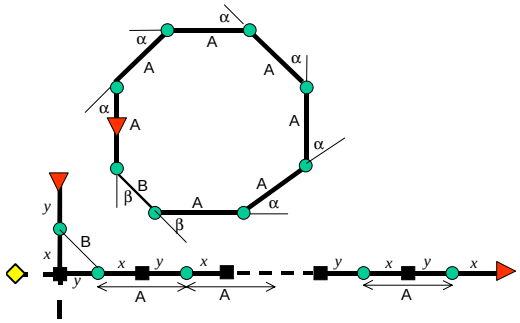

Figure 6: A near-regular-polygon docking technique

Consider a "tail-attaching" action that connects the tail of a scorpion to its leg (or the tail of a snake to its body). This action is a basic step in reconfiguration, such as turning a scorpion to a snake or vice versa. After the tail and the leg are connected, the leg can then be disconnected from the body by a "break-loop" action, thus becoming the tip module in the new tail. These actions can be repeated until all legs have been merged into the tail. Similarly, these two actions can also be used for a snake to convert its tail to multiple legs. The "tail-attaching" action can also allow a snake to connect its tail to its head to form a "hoop snake" or a "rolling track.."

In the classic inverse kinematics theory, this "tail-attaching" action is a hyper-redundant problem. To bring the tail to its docking position, some intensive matrix computation is performed to determine the bending angles for all the modules involved in the action. Multiple solutions do exist, so that the system must make a choice based on the domain knowledge. In the context of hormone-based control, however, we have developed a technique called "near-regular-polygon" to derive a solution that is efficient to compute and suitable for hormones.

The basic idea of this technique is illustrated in Figure 6. The bottom of the figure is a scorpion configuration, where each module is represented by a segment between two squares (the squares are joint connectors and the circles are DOF). The objective is to dock the tail connector (marked by a triangle) to a leg connector (another triangle). The problem is easier if the tail is to be attached to the head (marked by a diamond), for then the entire body could form a regular polygon with the length of edge equal to the length of a module ($A=x+y$). In this case, each DOF bends $(360/N)°$, where N is the number of modules in the loop. The situation of attaching the tail to a leg is more complicated, because a leg is perpendicularly attached to the body, so one of the edges in the final loop has a length $B=(x^2+y^2)^{1/2}$ and $B<A$. However, realizing that the final loop (illustrated with DOFs at the top part of Figure 6) is a near-regular polygon with equal-length edges except one, and the angles α and β have the relation $(N-2)\alpha + 2\beta = 360°$, we can search for the answers for α and β by a hill-climbing approach, as shown in Algorithm 2.

Let α=360/N and η=IR_Approximate_Sense(M_{tail}),

1. $\alpha = \alpha + \Delta$, /* Δ is a small constant */
2. $\beta = (360-(N-2)\alpha)/2$,
3. Move DOFs using α and β,
4. η' = IR_Approximate_Sense(M_{tail}),
5. If $\eta<\eta'$, then $\eta=\eta'$ and goto 1,
6. If $\eta\geq\eta'$, then stop.

Algorithm 2: Near-regular-polygon for tail-attaching

As we can see, this algorithm terminates when the IR approximate sensor value reaches a maximal point. Note that this is a closed-loop converging algorithm. In a self-reconfigurable robot where motors do not have high-precision controls, a closed-loop converging algorithm is much more reliable than an open-loop algorithm.

8. IMPLEMENTATION & EXPERIMENTS

Up to this point, we have given a primitive theory about hormone-based control. This section describes our current implementation of hormone-based control on the CONRO robots. As we have mentioned before, the computational resources of a CONRO module are very limited. The STAMP II micro-controller we are using has only 2K bytes of memory for BASIC programs, and 32 bytes of RAM space for variables. We have taken considerable effort to make the control program precise and powerful, yet as concise as possible.

All modules are loaded with the same program. This program performs the management functions for local devices, communications, and hormones. The local devices of a module include two servomotors, one connector-release mechanism, one analog position sensor, and four IR approximate sensors. A module has four bi-directional communication links, all using the same IR device as the approximate sensors. The communication uses the RS-323 serial protocol and shares two 4-byte software buffers due to the space limitation. Hormones and other messages are implemented as a 4-byte packet, which contains information about its type, sender, receiver, inlink, outlinks, action code, and lifetime (imagine that all these must be coded in 32 bits!). Since there is no interrupt mechanism in the micro-controller, the program must monitor all activities by timesharing. Furthermore, the program is also an interpreter for all hormones and messages. In the current implementation, there are 39 types of action codes, ranging from motor control to sensor readings, from hormone relay to synchronization. In some sense, all the semantics of hormones are rooted and implemented in this 2K-byte BASIC program. This success is largely due to the simplicity and powerful capability of the hormone-based control.

Up to now, we have developed two generations of CONRO prototypes. The first generation has 16 modules and they are tethered for external power and control. Using a centralized control program without hormones, the modules in a chain configuration can move as a caterpillar and can configure into a rolling track. In a hexapod configuration, the robot can move forward, backward and turn. The details of these experiments are reported in two separate papers [13, 14], and can be seen at our web site (http://www.isi.edu/conro).

The second-generation CONRO prototype has 20 autonomous modules, each equipped with on-board power, sensors, and micro-controller. (Figure 1 in this paper is from this generation). The software of these modules is an implementation of what has been described in this paper. In a snake configuration that has four modules, a CONRO robot can move forward with a speed of 1foot/minute using the caterpillar move hormones. In a scorpion configuration that has nine modules, a CONRO robot can push up on six legs and stand up by itself. We are currently making it walk, as a spider or a crab. In the experiments we have done so far, all hormones are triggered externally by human-generated messages. A host computer is tethered to one of the modules in the configuration, and a human interface on the host was developed to generate, send, and monitor hormones. We are currently working on migrating the hormone generation capability onto the modules and then running the modules without any external assistance.

As of the writing of this paper, we have not had time to conduct all the locomotion and reconfiguration described in this paper. Reconfiguration of the robots is still done manually. However, modules can automatically discover the current configuration topology once they are connected. When a robot starts running, every module will systematically probe its four communication links and query the neighbor module's ID if a link is determined to be active (i.e., it is connected to another module). The active neighbor information is recorded in the local memory, and can be exchanged if necessary. Modules can also report errors to the host computer. Typical errors include a breakdown of an active communication link (when there is no reply for a relay), or a feedback signal when a motor cannot reach its target position.

9. CONCLUSION & FUTURE RESEARCH

In this paper, we have presented a new approach for controlling self-reconfigurable robots using the biological concept of a hormone, and reported our current theory and implementation of this approach. Preliminary experiments on our existing CONRO reconfigurable robots have shown that this approach has very promising potential. It is our belief that nature has offered us a very simple and powerful mechanism for controlling distributed systems, and there is a great deal more to be learned from biological systems.

There are many future research directions we can take from here. In particular, we would like to explore the possibility of having hormones carry not only control signals, but also control programs to be executed at different sites. Furthermore, we have only scratched the surface of hormone management in a distributed network of autonomous systems. New methods must be developed for generating the right hormones at the right time, and for collaboration and negotiation between simultaneously active hormones. We will continue to exploit the new opportunities made possible by our CONRO prototype robots in order to develop a complete theory of hormone-based control. Since self-reconfigurable robots share so many important features with distributed autonomous systems/agents in general, we expect such a theory will have a much broader range of applications than simply robot control.

10. ACKNOWLEDGMENTS

We are grateful to our CONRO colleagues for their moral and technical support at various times. In particular, we thank Andres

Castano, Robert Kovac, Ramesh Chokkalingam, and Sunan Tugsinavisut for building the robot; Alberto Behar for his work on the miniature camera; and Behrokh Khoshnevis for the initial design of the connector. We also thank Paul Rosenbloom for suggesting the relationship between hormones and marker-passing systems. 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.

11. REFERENCES

1. Kotay, K., D. Rus, M. Vona, and C. McGray. *The self-reconfiguring robotic molecule*. in *Proceedings of IEEE International Conference on Robotics and Automation*. 1998.

2. 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.

3. Yim, M., *Locomotion with a unit-modular reconfigurable robot (Ph.D. Thesis)*, in *Department of Mechanical Engineering*. 1994, Stanford University.

4. Yim, M., D.G. Duff, K.D. Roufas. *PolyBot: A Modular Reconfigurable Robot*. in *Proceedings of the IEEE International Conference on Robotics and Automation*. 2000.

5. Fukuda, T., and Y. Kawauchi. *Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator*. in *Proceedings of IEEE International Conference on Robotics and Automation*. 1990.

6. Fujita, M., H. Kitano and K. Kageyama. *Reconifurable physical agents*. in *International Conference on Autonomous Agents*. 1998.

7. Fahlman, S., *Three Flavors of Parallelism*, . 1982, Carnegie Mellon Universtiy: Pittsburgh.

8. Arkin, R.C., *Homeostatic Control for a Mobile Robot: Dynamic Replanning in Hazardous Environments.* Journal of Robotic Systems, 1992. **9**(2): p. 197-214.

9. Murata, S., H. Kurokawa, E. Toshida, K. Tomita, and S. Kokaji. *A 3-D self-reconfigurable structure.* in *Proceedings of the IEEE International Conference on Robotics and Automation*. 1998.

10. Yoshida, E., S. Murata, K. Tomita, H. Kurokawa, and S. Kokaji. *Distributed formation control of a modular mechanical system*. in *Proceedings of the International Conference on Intelligent Robots and Systems*. 1997.

11. Paredis, C., and P. Khosla. *Design of modular fault tolerant manipulators*. in *Proceedings of the First Workshop on Algorithmic Foundations of Robotics*. 1995.

12. 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.

13. Will, P., A. Castano, W.-M. Shen. *Robot modularity for self-reconfiguration*. in *Sensor Fusion and Decentralized Control in Robotic Systems II*. 1999.

14. Castano, A., W.-M. Shen, P. Will, *CONRO: Towards Miniature Self-Sufficient Metamorphic Robots.* Autonomous Robots (to appear), 2000.