Global Behavior via Cooperative Local Control

Cynthia Ferrell *
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square, Room 741
Cambridge, MA 02139
(617)253-0997
fax: (617)253-0039
ferrell@ai.mit.edu

*Support for this research was provided in part by a NASA Graduate Student Researcher Program Fellowship administered through the Jet Propulsion Laboratory, by Jet Propulsion Laboratory grant 959333, and by the Advanced Research Projects Agency under Office of Naval Research contract N00014–91–J–4038.

Abstract

The purpose of this paper is twofold. First, we outline important issues in designing real-time controllers for robots with numerous sensors, actuators, and behaviors. We address these issues by implementing a behavior based controller on a sophisticated autonomous robot. Hence, this work provides a point of reference for the scalability, ease of design, and effectiveness of the behavior based control for complex robots. Second, we explore the viability of using cooperation among local controllers to achieve coherent global behavior. Our approach is to decompose a difficult control task for a complex robot into a multitude of simpler control tasks for robotic subsystems. We illustrate and examine the effectiveness of this approach via rough terrain locomotion using an autonomous hexapod robot. Traversing rough terrain is a good task to test the viability of this approach because it requires a considerable amount of leg coordination. We found that implementing a complicated global control task with cooperating local controllers can effectively control complex robots.

1 Introduction

As the field of autonomous robot control advances, people will continue to design robots with greater capabilities to perform more challenging and comprehensive tasks. For example, research is underway to develop autonomous systems capable of exploring other planets (Angle & Brooks 1990), (Bares & Whittaker 1990) and deep within oceans (Payton, Keirsey, Kimple, Krozel & Rosenblatt 1992), (Stuart 1988). Equipping robots with more sensors increases the quantity and reliability of information the robot can extract from its environment. Subsequently, the robot can use this information to behave more intelligently. Providing robots with more actuators increases the physical capabilities of the robot (i.e. arms, hands, improved locomotion, active vision). Programming the robot with more behavioral capabilities makes it more flexible and versatile. However, expanding the capabilities of robots also makes them more complicated. Understanding how to control robots of significant complexity will be an important issue in the future.

This paper explores issues and methods which we encountered for designing and implementing controllers for complex robots with many sensors and actuators. We outline these important issues, and explore the viability of using cooperation among local controllers to achieve coherent global behavior. This approach decomposes a difficult global control task of a complex robot into a multitude of simpler local control tasks for robotic subsystems. The approach is inspired by the apparent decomposition of locomotion control in insects. We illustrate and examine the effectiveness of this approach via rough terrain locomotion using an autonomous hexapod.

2 Hannibal

Hannibal (shown in figure 1), is the small autonomous robot used in our experiments. It was designed and built under the supervision of Prof. Rodney Brooks in the Artificial Intelligence lab at MIT (Angle & Brooks 1990) as an experimental platform to explore planetary micro-rover control issues (Angle 1991).

2.1 Size

Hannibal benefits in several ways from its small size as argued in (Angle 1991). Hannibal measures 14 inches long, stands eight inches high, and weight 6 pounds. Because Hannibal is small, it has relatively low mass which gives rise to several advantages, including reduced dynamic effects, which simplifies control. Another advantage is lower power consumption–a smaller robot can be driven with smaller, lower power motors. The greatest advantage is a favorable strength-to-weight ratio of the robot's structure. The strength of a structure scales by its cross sectional area, while the weight of a structure scales by its volume. As a structure is proportionally scaled up, its weight increases at a faster rate than its strength. Hence, it is relatively easy to make a small structure quite strong with little mass.

2.2 Mechanics

When designing the robot, careful consideration was given to mobility, sensing, and robustness issues. Much has been said concerning the advantages of legged vehicles over wheeled vehicles in regard to their mobility over rough terrain (Song & Waldron 1989), (Hirose 1984). Hannibal was engineered with six 3 degree of freedom (DOF) legs and a 1 degree of free-

dom body to give Hannibal enough degrees of freedom to traverse rough terrain. Each of Hannibal's six legs has 3 DOFs to allow arbitrary foot placement on a surface. The *lift axis* is horizontal, and a rotational actuator raises and lowers the foot by rotating the leg about this axis. The *swing axis* is vertical which is important for the efficiency of the robot. When the leg makes a step, most of the motion of the leg is rotation about the swing axis and therefore parallel to gravitational forces. As a result, relatively little work is done against gravity as the leg swings through the environment. A rotational actuator advances the leg along the direction of travel by rotating the leg about the swing axis. The *extension axis* is orthogonal to the lift and swing axes. A linear actuator extends and contracts the leg along this axis. Finally, a global degree of freedom mounted on Hannibal's body links the rotation of all six legs about their axles together. A spine actuator is responsible for rotating the legs about this global degree of freedom. The purpose of this DOF is to ensure the legs are always vertical. Angle & Brooks (1990) argues that the load on each of the leg motors is independent of robot inclination if the legs are always vertical. As the robot's inclination increases during a climb, the global rotation of the legs brings the center of mass of the robot closer to the surface being climbed, and keeps it within the polygon of support for any inclination.

2.3 Sensors

Rovers must have sufficient terrain sensing capabilities to locomote safely and effectively over rugged terrain. To meet this requirement, Hannibal's legs are encrusted with a multitude of sensors that provide complementary information. This serves several purposes. First, multiple sensors provide the robot with more information about its environment, which

helps the robot behave more intelligently. Second, multiple complementary sensors increase sensing robustness–if one sensor fails, its complementary sensors can provide the robot with similar information. Third, they enhance sensing reliability since the information from each sensor can be used to confirm the results of the other complementary sensors. This increases the confidence in the net sensor output. Ferrell (1994) describes how complementary sensor suites can be used to integrate fault tolerance capabilities into distributed controllers.

Hannibal receives a tremendous amount of sensor information. The robot has over 60 sensors of 5 different types. These sensors are used to sense the immediate terrain, provide the robot with knowledge of its physical configuration, and servo the motors which control its DOFs to specified positions. Most of Hannibal's sensors are mounted on its legs. These sensors provide the robot with terrain information as the legs sweep through the environment. Strain gauges mounted on the legs measure the force the external world is exerting on them. Hannibal uses them to sense vertical loading of the legs and to detect collisions the legs suffer as they move through the environment. An ankle potentiometer on each leg is used to sense foot loading. The robot uses this information to find secure footholds as it walks through the environment. Joint potentiometers are used to measure joint angles of all the robot's degrees of freedom. This information is used to servo control the motors to specified positions and to inform the robot of its physical configuration. Joint velocity sensors are used to control the motors. Velocity information in conjunction with target position information can be used to roughly sense leg collisions. For example, if a leg prematurely stops before it reaches its target position, the leg may have experienced a collision along the way. An inclinometer is used to sense the robot's pitch and roll. The spine potentiometer and the inclinometer, both mounted on Hannibal's body, are used to control Hannibal's spine actuator to keep all the

legs vertical.

2.4 Computing

Not surprisingly, Hannibal is quite complex for its size. It is approximately the size of a bread box and uses over 60 sensor signals to orchestrate 19 DOFs while locomoting over rough terrain. To make the design and control of this robot manageable, Hannibal was divided into smart subsystems which are linked together with a medium speed serial network (I^2C bus).

Hannibal is organized with a single central processor to which seven subsystem control processors are attached. Each leg and the body are individual subsystems that posses a set of sensors, actuators and a satellite processor. This allows high bandwidth communication and motor servoing to be handled locally, with only high level communication on the serial network.

- The master processor is a 15 MHz Signetics 68070. It serves as the computational engine of the robot and runs the behavior control code. It receives sensor information from the robot's subsystems over the serial bus and uses this information to intelligently control the robot's behavior. It dictates the actions the robot takes by sending position, velocity, and force control commands to the actuators of the subsystems.

- Each local satellite processor is a Signetics 87c751. The processor has two responsibilities. The first is to acquire the sensor information of its subsystem and send it to the master processor. The second is to receive commands from the master processor to servo the actuators of its subsystem. Servo control code running on this processor

at 100 Hz moves the subsystem's actuators to the commanded position with velocity and/or force considerations. The processor uses the command information from the master processor and the sensor information of its subsystem to generate the pulse width modulation wave forms which drive the actuators.

- The I^2C serial bus is a 100 K bit communication system which is used to connect the various subsystems on Hannibal. Bus transactions occur 30 % of the time which allows the satellite processors to exchange information with the master processor at 10 Hz. The bus is idle for 70 % of the time to allow the satellite processors to execute their servo control code. The I^2C bus is not fast enough to do real-time motor servoing, and this limitation forces modularity on the robot.

2.5 Software

Hannibal is programmed using the *subsumption* approach, a set of philosophical concepts about robot behavior design which stresses the issues of reactivity, concurrency, and real-time control. Brooks (1986) proposed the idea of subsumption as an alternative approach to the robot behavior control problem. This approach decomposes the control problem into task achieving layers. Each slice in the vertical division is level of competence. The main idea is that we can build layers of a control system corresponding to each level of competence and simply add a new layer to an existing set to move to the next higher level of overall competence. The lower layers run continually and are unaware of higher layers. However, when the higher layers wish to take control, they can subsume the roles of lower levels. This approach is robust since failure of any layer does not affect the layers below. Additionally, this

organization allows for modular addition and removal of behaviors, and thus for incremental design and debugging. Most importantly, it allows for a tight loop between sensing and action which can be performed quickly and with much less computation.

The subsumption approach is embodied in the *Subsumption Architecture* which uses networks of finite state machines augmented with timing elements and registers (*AFSMs* or *processes*) to construct simple rules. The networks can be thought of as explicit wiring diagrams connecting the outputs of AFSMs to the inputs of AFSMs. Message passing between AFSMs is performed by connecting a wire to the input of a AFSM and writing to its register. On Hannibal, messages are 16 bit integer values. The internal state of an AFSM can change either by the expiration of a timer or by the arrival of an incoming message. In this paper, we say a process *excites* another when a message passed from the first AFSM is received by the second and causes the second to become active (perform computation, command actuators, or send output messages). As new wires are added to the network they can connect to existing registers, *suppress* inputs (a message from an AFSM stops all inputs along a wire to another for a fixed time period), or *inhibit* outputs (a message from an AFSM stops all outputs along a wire from another for a fixed time period). In this paper, we define combinations of AFSMs as either *behaviors* if their processing results in an outwardly observable affect, or *agents* if their processing has only an internal affect. Behaviors and agents are the building blocks of the *Behavior Language* (Brooks 1990).

Hannibal is programmed in Behavior Language, a high level language for writing subsumption programs. Behaviors run concurrently (simulated concurrency if running on a single processor) and asynchronously, perform their own perception, monitor their input wires, perform computation, control actuators, or send messages through their output wires.

It is important to design the controller such that conflicting behaviors are not active at the same time. To deal with this situation, behaviors have the ability to inhibit outputs or suppress inputs of other behaviors. Inhibition of outputs is used when the inhibiting behavior does not want the inhibited behavior's outputs influencing the system. Behaviors can prevent other behaviors from becoming active by suppressing inputs used for activating other behaviors. Individual behaviors are connected to form task-achieving modules, and these task-achieving modules can be grouped together to form layers. Each layer is a level of competence which corresponds to the robot's abilities.

3 Controller Design Issues

Hannibal is a complex robot which operates in a complex environment. Naturally occurring terrain has holes, cliffs, obstacles of various sizes, and undulations. To locomote over varying terrain, the robot needs sufficient sensory information to recognize changes in the terrain. Hannibal's 60-plus sensors provide the information necessary to negotiate obstacles and avoid hazards. Managing Hannibal's sensors and actuators and controlling the robot in real-time are important for the robot's success and safety. To complete this formidable task, Hannibal's controller must satisfy several requirements:

- The controller must *scale* well to efficiently and effectively utilize numerous sensors, actuators, and processes. Adding more sensors, actuators, or processes to the system increases the load on the controller. We do not want this additional load to result in computational bottlenecks. If the controller scaled poorly, attempting to enhance the robot's capabilities by adding more sensors, actuators, or processes could degrade the

robot's performance rather than enhance it.

- The controller must be *modular* so that additional sensors, actuators, or processes can be integrated readily. Specifically, we want to be able to quickly add or change the processes which interpret new sensor information, utilize new actuators, or add new processing capabilities to the system.

- The controller must be *flexible*. It must perform diverse functions such as characterizing terrain features and controlling the robot through a variety of maneuvers.

- The controller must be *robust* and *adaptive*. The robot's behavior should be resilient to both environmental changes and internal changes. Environmental changes are attributed to irregular terrain. The robot must be able to detect these terrain variations and adapt to them accordingly. Internal changes are attributed to various hardware failures, erroneous sensor readings, or sensor drift. The robot must recognize these faults and minimize their effect on the robot's performance (Ferrell 1993).

Numerous papers argue in favor of behavior based control for modularity, flexibility, robustness, and adaptability considerations (Brooks 1986), (Maes 1990), (Rosenblatt & Payton 1989). These qualities are important as more capabilities are added to controllers. To date, the subsumption approach has been successfully demonstrated on several autonomous robots in our lab such as Toto (Mataric 1990), Herbert (Connell 1989), Squirt (Flynn, Brooks, Wells & Barrett 1989), and Genghis (Brooks 1989). These robots are representative of the typical level of complexity of other behavior based autonomous robots in the field.

Hannibal's controller is significantly more complicated than the controllers of these earlier systems. Sensori-motor integration on Hannibal is more involved than these earlier systems

because Hannibal has considerably more sensors and actuators. Hannibal also has a comparatively large behavior repertoire including insect-like gaits and rough terrain locomotion as well as extensive failure recognition, fault tolerance, and sensor processing capabilities (Ferrell 1993), (Ferrell 1994). The fully compiled system consists of approximately 1500 AFSMs performing diverse tasks. Hence, we view this work as a point of reference for the scalability, ease of design, and effectiveness of the behavior based approach for addressing the control of increasingly more complex robots.

4 Local Control with Cooperation

This section presents a method for controlling a complex robot in real-time by efficiently managing its numerous sensors and actuators. In this method, coherent global behavior is achieved by local controllers that cooperate with each other. The idea is to simplify the control problem by decomposing the global task into local tasks for the robot's subsystems. As described earlier, each subsystem has its own set of sensors, actuators, servo code. By writing behavior code for each subsystem, each subsystem can be treated as a *sub-robot* (recall that the behavior control code is distributed among the subsystems in software since all behavior control code runs on one processor.). The sub-robots are significantly simpler than the overall robot because each subsystem has significantly fewer sensors and actuators and performs relatively simple tasks. Since each local controller is responsible for managing a simpler system, the load on each local controller is reduced accordingly.

The local controllers must cooperate for the overall system to have coherent behavior. To keep the load on any one controller to a minimum, we establish an abstraction barrier

between low level control issues (processing sensor information and commanding actuators) and higher level control issues (coordinating the behavior of the subsystems). On one side of the abstraction barrier, the local controllers are responsible for implementing tight coupling between sensors and actuators to achieve real-time behavior control of the subsystems. On the other side, the local controllers communicate with each other to coordinate their behavior. This is done either directly between local controllers, or indirectly through special *broadcast* agents which communicate a message to all relevant subsystems. All communication is implemented using the message passing mechanisms as described in section 2.5. The messages sent between local controllers excite or inhibit agents within the local controllers. Hence, these messages coordinate the behavior of the subsystems without having to explicitly deal with sensory information and commanding actuators.

To simplify controlling a robot of Hannibal's complexity, we decomposed the global locomotion problem into several local problems by implementing a behavior-based controller for each subsystem (such as the head, the body, and each leg) which is responsible for governing the behavior of that subsystem. Instead of controlling a complex robot with a single behavior-based controller, the task is distributed among several controllers–each responsible for a subset of the overall control problem. Thus, Hannibal's control structure is distributed on two levels: each local controller is distributed among behaviors, and the overall control is distributed among several behavior-based controllers.

With this approach, Hannibal can be modeled as a robot which is composed of several sub-robots. Each of Hannibal's subsystems (legs, head, and body) is an autonomous sub-robot possessing its own set of sensors, actuators, servo control, and behavior control. However, these sub-robots cannot function independently because they are physically coupled to each

other through the robot and the terrain. Consequently, they must cooperate so the overall system can achieve its goals. To achieve cooperation the sub-robots communicate with each other, with direct messages or through special broadcast agents.

5 Insect-Inspired Locomotion

The local control with cooperation approach decomposes a difficult global control task of a complex robot into a multitude of simpler local control tasks for robotic subsystems. The approach is inspired by the apparent decomposition of locomotion control in insects. Several researchers have implemented biologically motivated locomotion controllers, either on robots or in simulation, to understand neural control mechanisms and to find better ways of programming six legged robots. A variety of insect motivated approaches have been used to produce real-time locomotion with limited computational power. This is particularly important for small autonomous hexapod robots that carry their own computation. In this section, we review four insect-like locomotion controllers. We compare these implementations to Hannibal's controller in section 12.

5.1 Definition of terms

Below are several terms we use throughout this section. Please also refer to figure 2.

1. *Protraction*: The leg moves towards the front of the body.

2. *Retraction*: The leg moves towards the rear of the body.

3. *Power stroke*: The leg is on the ground, where it supports and propels the body. In forward walking, the leg retracts during this phase. Also called the stance phase or the support phase.

4. *Return stroke*: The leg lifts and swings to the starting position of the next power stroke. In forward walking, the leg protracts during this phase. Also called the swing phase or the recovery phase.

5. *Anterior extreme position (AEP)*: In forward walking, this is the target position of the swing degree of freedom during the return stroke.

6. *Posterior extreme position (PEP)*: In forward walking, this is the target position of the swing degree of freedom during the power stroke.

5.2 Cruse

Cruse (1994) describes a neural net implementation of a locomotion controller modeled after the neural control of walking stick insects (Cruse 1980). The controller is implemented in simulation where an agent with walking-stick insect mechanics locomotes over flat or irregular terrain and can be subjected to various leg perturbations. The simulated system can walk at different speeds using different gaits, walk over irregular surfaces composed of small obstacles, and quickly re-establish gait coordination in response to leg perturbations. However, the simulation does not take loading considerations into account, so it is unclear how this simulation would transfer to a physical system.

In the controller, each leg has its own control system that generates rhythmic stepping movements. Each of these local controllers is a neural network consisting of three sub-

networks: a superior state selector net and subordinate swing and stance nets. The state selector net (consisting of two input units and two output units) governs the transition between the power stroke and the return stroke of the leg by rhythmically suppressing the output of either the swing network or the stance network. The swing network (consisting of 6 input units, 3 output units, and a single bias unit) was trained to produce return stroke movements despite leg perturbations. The stance network (consisting of two inputs and six outputs) was trained to produce the power stroke movement.

The state selector net was trained to switch between the stance network and the swing network based on sensory input and the condition of adjacent legs. The state selector net causes a transition from the swing phase to the stance phase when ground contact is achieved, and it causes a transition from the stance phase to the swing phase when the leg reaches the PEP position threshold. In the simulation, three mechanisms modify the PEP threshold: 1) A rostrally directed inhibitory signal sent during the return stroke of the next caudal leg. 2) A rostrally directed excitatory signal sent when the next caudal leg and contra-lateral leg begin active retraction. 3) A caudally directed influence which is dependent on the position of the next rostral leg and the contra-lateral leg. All other connections responsible for coordinating influences were hard-wired based on biological experiments with walking stick insects. Several gaits emerge from the cooperation of local leg controllers where their interaction is governed by these mechanisms.

5.3 Case Western Hexapod

Beer and colleagues implemented a neural network control architecture on a small walking robot. The robot measures 50 cm long, 30 cm wide, weighs 1 kg and has six 2 DOF legs (Beer

& Chiel 1993), (Chiel, Quinn, Espenschied, & Beer 1992), (Quinn & Espenschied 1993). The neural controller was developed by Beer and was inspired by Pearson's *flexor burst-generator* model of cockroach locomotion. Using this controller, the Case Western Hexapod is capable of producing a continuous range of wave gaits on flat terrain by varying the tonic level of activity of the command neuron. The robot was not tested under conditions that would effect leg loading such as walking it over irregular terrain. However, the system is robust to lesion studies performed by removing certain sensors or controller connections.

The controller is implemented with 37 artificial neurons: six neurons per local leg controller and one command neuron. In this model, two mechanisms generate the stance/swing cycle of each leg, one based on rhythmic outputs of a pacemaker and the other on sensory input. At the center of each leg controller is a pacemaker neuron whose output rhythmically oscillates. A pacemaker burst initiates a swing by inhibiting the foot and backward swing motor neurons and exciting the forward motor neuron. This causes the foot to lift off the ground and the leg to swing forward. Between pacemaker bursts, the foot is down and tonic excitation from the command neuron moves the leg backward. On each leg, the output of a pacemaker neuron is tuned by feedback from two sensor neurons that signal when the leg is reaching the AEP or the PEP. Approaching the AEP encourages a pacemaker neuron to terminate a burst by inhibiting it. Approaching the PEP encourages a pacemaker neuron to initiate a burst by exciting it. Inserting mutually inhibitory connections between the pacemaker neurons of adjacent legs generates statically stable gaits, and phase-locking the pacemaker neurons on each side of the body enforces metachronal waves.

5.4 SSA Hexapod

In the early 80s, Marc Donner implemented an insect-inspired locomotion controller on the SSA Hexapod (Donner 1987). The SSA is a large hexapod built at CMU by Dr. Ivan Sutherland of Sutherland, Sproull, and Associates. The robot has a 100 inch steel frame, weighs 1800 pounds, and can carry a human operator. Its six 3 DOF legs are hydraulically actuated and have position and force sensing for each DOF (among other sensors). The controller implements a statically stable, metachronal gait.

The locomotion part of the controller can be described by approximately 26 concurrent processes. Each local controller uses four processes that move the leg through a step cycle. The step cycles of the legs are coordinated by constraining when the stance-to-swing transition of each leg occurs. These constraints are provided by inhibition and excitation signals sent between adjacent local leg controllers. The stance-to-swing transition of a given leg occurs when the measured leg load is less than the *unload decision threshold* for that leg. Excitation and inhibition mechanisms were used to modulate this decision threshold in three ways: 1) Each swinging leg inhibits its neighbors from swinging by subtracting a constant from the neighbors' unload decision threshold. 2) Each supporting leg excites its forward neighbor to swing by adding an excitatory constant to the neighbors' unload decision threshold, and removes this constant when the supporting leg begins its swing phase. 3) When a rear leg reaches the halfway point of its drive stroke, it excites its crosswise neighbor to swing by adding an excitatory constant to its crosswise neighbor's unload decision threshold (the local controllers of the rear legs have an extra process to implement this mechanism). The first mechanism maintains gait stability, the second mechanism encourages a back-to-front

wave gait, and the third mechanism encourages proper left-to-right side phasing. Statically stable gait coordination emergs from the interaction of these mechanisms.

5.5 Genghis

Genghis is a small autonomous walking robot built in the mid '80s by the Mobile Robotics Group at MIT (Brooks 1989). It measures 35 cm long, 25 cm wide, weighs 1 kg, and has six 2 DOF legs. Using the Subsumption Architecture, a network of 57 AFSMs was designed which allows Genghis to walk with a ripple gait, traverse rough terrain, and follow people.

Basic walking on flat terrain is implemented by 32 AFSMs. Two centralized AFSMs are used for global coordination. The *Walk* machine acts as a centralized gait sequencer by telling each leg when to begin its swing phase (only a single gait can be implemented at a time). The centralized *Alpha Balance* machine determines how much each supporting leg propels the robot forward. By default, it continually sends motor commands to all the advance motors. Each local leg controller consists of five AFSMs and is responsible for generating the cyclic motion of that leg. Two of these machines, *Beta Pos* and *Alpha Pos* simply send commands to the lift and advance motors, respectively. The remaining three machines produce the recovery movement. The *Leg Down* machine continually tells the Beta Pos machine to put the leg down. Hence, the default position is for the leg to support the robot. This message gets through to the Beta Pos machine except when the Up Leg Trigger machine blocks it. Once activated by the Walk machine, the *Up Leg Trigger* machine inhibits this message for a predetermined period of time and lifts the leg. Once the leg is off of the ground, the *Alpha Advance* machine commands the Alpha Pos machine to swing the leg forward while inhibiting the Alpha Balance machine. Hence, whenever the leg is lifted, it is

reflexively swung forward as well. The leg lowers to support the body once the activation period of the Up Leg Trigger machine expires. The stance phase resumes when messages from the Alpha Balance machine are no longer inhibited by the Alpha Advance machine. By adding four more AFSMs to each leg, Genghis walks over obstacles. By adding an extra centralized machine, Genghis turns in either direction.

6 Basic Walking

This section illustrates the local control with cooperation method by presenting Hannibal's basic walking implementation. Locomotion control is distributed evenly among the six legs. Each local leg controller is responsible for generating the cyclic motion of its leg. The local leg controllers run simultaneously; however, they are not independent of one another. The six legs must work together as a team for the robot to move effectively. Using this approach, we have implemented the following basic locomotion capabilities on Hannibal (Ferrell 1993):

- Walk in a statically stable manner
- Change speed by switching among a variety of insect-like gaits (wave gaits)
- Change direction of travel
- Turn with various radii of curvature
- Compensate for broken legs

For space limitations, we only present the implementation of the first two capabilities in this paper. The design of the corresponding behaviors is inspired by the work of Keir

Pearson and his collaborators. They investigated the neural systems that control walking in the cockroach (Wilson 1966), (Pearson 1976) and developed neurological models to explain the control of an individual leg and the coordination between legs.

6.1 Step Cycle Generation

In Pearson's model for individual leg control, a pacemaker on each leg is responsible for generating the cyclic motion of the leg (see figure 3). It does this by coordinating the return stroke and power stroke of the leg. Similarly, each of Hannibal's legs has an *oscillator agent* which generates the cyclic motion of the leg. The oscillator agent is modeled as a clock that cycles through its values at regular time steps (see figure 5). A *stage* spans one oscillator clock period. We say the oscillator period (time between adjacent oscillator peaks) spans M clock stages. Each oscillator is synchronized to a common clock. In the current implementation, the oscillator clock value changes every 0.6 seconds.

To produce the return stroke, the oscillator agent excites a *lift agent* (which lifts the leg), a *swing agent* (which swings the leg to the AEP), and a *step agent* (which lowers the leg until it supports the body) as shown in figure 4. The lift, swing, and step agents serve a similar function as the flexor neurons of Pearson's model. The lift portion of the recovery phase is performed during the first stage of the oscillator (clock value = 1). The step portion of the recovery phase is performed during the second stage of the oscillator (clock value = 2). During these two stages, the leg swings to the AEP.

To produce the power stroke, the oscillator agent excites the *support agent* as shown in figure 4. The support agent serves a similar function as the extensor neurons of Pearson's model. For clock stages 3 through M, the support agent moves the leg an incremental

amount towards the PEP each clock stage, where the increment is equal to S/N where S = abs(PEP - AEP) and N = number of support stages. This increment is determined so that the leg reaches the PEP during the last of the series of support stages. Ergo all supporting legs propel the body in synchrony, and the duration of the power stroke is equal for all the legs. The transition from the support phase to the swing phase occurs on clock value = 1, immediately after the leg reaches the PEP.

6.2 Gait Generation

Similar to Pearson's model, Hannibal changes gait as a function of oscillator frequency. A broadcast agent called the *speed agent* (analogous to Pearson's command neuron) is responsible for changing the leg oscillator frequency of all the leg oscillators. In addition, local inter-leg communication agents called *coupling agents* are responsible for establishing the phasing between leg oscillators by sending coupling signal messages between local leg controllers . Changing the value of M changes the oscillator frequency. This excites the coupling agents to change the phase between the metachronal waves along each side of the body (figure 6), which causes different wave gaits to emerge.

Figures 7 and 8 illustrate an example of the networks responsible for gait transitions. Currently, Hannibal uses three gaits: a slow wave gait, a ripple gait, and a tripod gait. Wilson (1966) reports the slow wave gait is the slowest gait and the tripod gait is the fastest gait observed in insects. More gaits could easily be implemented using this network by varying M and the phasing between leg oscillators.

- The Speed agent: Whenever Hannibal wants to change speed, this broadcast agent

sends a new M value to all the oscillators. This serves to extend the support phase of the step cycle since the recover phase takes a constant amount of time independent of walking speed. Higher level agents command speed through this agent.

- The Activate-Phasing agent: If a new M value is sent to the leg oscillators from the Speed agent, this broadcast agent finds the furthest posterior leg performing the lift-and-swing stage (the most rearward leg with its oscillator in the first stage) and activates its coupling agent.

- Coupling agents: When a different M value is sent to the leg oscillators, the instantaneous oscillator clock values must be re-phased with respect to each other to maintain a proper gait. The active coupling agent sends messages to all legs which re-initialize their current oscillator clock values (with respect to the oscillator clock value of the leg with the active coupling agent). Correspondingly, the leg oscillators simply activate the proper agent (lift, swing, step, or support) in accordance to its new clock value, and continue to run as usual. Figure 9 shows a Behavior Language code segment implementing the coupling agent for the left rear leg. Figures 7 and 8 illustrate this code segment.

6.3 Results: Flat Terrain Locomotion

Using this approach, Hannibal walks in real-time in a stable manner. The emergent gaits of this network posses similar characteristics to insect gaits (Wilson 1966), (Cruse 1990).

In regard to individual leg control:

- The oscillator agent keeps protraction time constant and decreases retraction time as

step frequency increases.

- The leg reaches the AEP when transitioning from the return stroke to the power stroke.
- The leg reaches the PEP when transitioning from the power stroke to the return stroke.

In regard to the coordination between legs, the leg oscillators are synchronized such that:

- A wave of protraction runs from posterior to anterior.
- No leg protracts until the leg behind is placed in a supporting position.
- The supporting legs propel the body in synchrony.
- The duration of the power stroke is the same for all legs.
- Contra lateral legs of the same segment are 180 degrees out of phase.
- On each side, the time between steps of the hind leg and middle leg and the time between steps of the middle leg and foreleg are constant.
- The time between the foreleg and hind leg steps varies inversely with frequency.

The transition between Hannibal s gaits is smooth and immediate. Figure 10 shows run time data of these three gaits, and figure 11 shows transitions between various gaits. These results demonstrate that local control with cooperation can be used to effectively control robotic subsystems in real-time and coordinate the behavior between robotic subsystems.

7 Rough Terrain Locomotion

Hannibal's rough terrain implementation highlights the different ways the local controllers communicate with each other. The task of traversing rough terrain can be viewed as a team effort where the legs must work together for the global system, Hannibal, to accomplish its task. Rough terrain control is fully distributed and exploits local control of the legs with inter-leg cooperation. Each leg is programmed to operate as a individual subsystem. Local leg control is responsible for handling challenges that confront individual legs such as stepping over an obstacle or finding a foothold. Effective inter-leg cooperation is vital because the legs are physically connected through Hannibal's body and the terrain. Legs communicate with each other not only to coordinate and synchronize their behavior, but also to alert each other of hazards and to recruit the help of the other legs when necessary. Using this approach, we have implemented the following capabilities on Hannibal (Ferrell 1993): (For space limitations, we do not present all of these capabilities in this paper.)

- Walk over small and medium sized obstacles
- Avoid large obstacles blocking the robot's path
- Search for footholds
- Walk over holes
- Avoid cliffs
- Adapt gait to terrain roughness
- Adapt to slopes

7.1 Inter-leg Communication

Negotiating obstacles exploits different uses of inter-leg cooperation, which is achieved through inter-leg communication. Hannibal's rough terrain network implements several types of inter-leg communication using the mechanisms provided by the Behavior Language (as described in section 2.5). For example, if a leg is stepping in a hole or over an obstacle, it tells the other legs to pause while it negotiates the hazard. It communicates this by having its step agent inhibit the oscillators of the other legs until it achieves ground contact (see figures 12 and 13). This communication ensures the robot's stability by delaying the next step cycle until all recovering legs support the body. A leg can also recruit help from other legs. For instance, when a leg is trying to step over an obstacle, it asks the supporting legs to raise the body to help it clear the obstacle. It requests this help by sending a message to activate the lift-body behaviors in the supporting legs (see figure 14). A leg can also alert the other legs of a dangerous situation. For example, if a leg either is stepping over a cliff or encounters an obstacle too large to step over, it tells the other legs to move the robot backwards, by sending a message to the direction and turn agents. A broadcast backup agent coordinates the direction and turn behaviors of all the legs (figure 15).

Communication between locally controlled legs is essential for the legs to maintain a cohesive effort. This is especially true if different legs want the robot to do different things. As shown in figure 16, one leg may hit a large obstacle and want the robot to back up, whereas another leg may be searching for a foothold and want the robot to hold still until it finds one. This is essentially behavior conflict, but on a larger scale than is typically encountered by other robots. Instead of having conflicting behaviors in competition over one

actuator, conflicting behaviors on different legs are in competing over the action of all the legs. An inter-leg priority scheme, implemented using inhibition and suppression mechanisms as described in section 2.5, is used to resolve inter-leg conflicts. For this example, the stability of the robot has the highest priority, so the legs wait until the searching leg finds a foothold before performing the backup maneuver.

8 Tests

We designed several tests to examine the effectiveness of different types of inter-leg communication for negotiating hazards and for dealing with behavior conflicts between legs. The success of Hannibal's rough terrain behaviors indicate the viability of inter-leg communication in achieving coherent behavior of the legs.

8.1 Individual obstacle tests

These tests challenged Hannibal's ability to handle a single obstacle in its path. The tests were designed to test the performance of specific rough terrain behaviors. They were used to determine if Hannibal could correctly characterize the type of obstacle in its path and respond to it. To isolate the robot's performance with respect to a particular hazard, the test terrain was flat with the exception of the terrain feature. Only one leg encountered the challenge at a time. The following terrain features were tested:

- Obstructive objects of varying heights and shapes
- Cliffs

- Holes or gaps in the terrain

8.2 Multiple obstacle tests

We also conducted tests to challenge Hannibal's ability to handle multiple obstacles in its path. The hazards could be of the same type or of different types, and either sequential or concurrent. The tests were designed to test the flexibility of the controller, Hannibal's ability to adapt to changing circumstances, the legs' ability to handle concurrent obstacles, and the legs' ability to behave in a unified manner–particularly when behaviors on different legs may be in conflict. We chose a few sample terrains, listed below, to test various capabilities (other terrains were used as well but served redundant purposes):

- *Small plateau*: This terrain causes the robot to frequently encounter cliffs while walking forwards, backwards, or turning. The terrain tests the ability of the robot's legs to maintain a cohesive effort since behaviors on different legs want the robot to move in different directions.

- *Multiple small obstacles*: This terrain causes the robot's legs to traverse obstacles concurrently. It tests the robot's ability to handle local terrain features simultaneously.

- *Crevice*: This terrain causes the robot's legs to sequentially step over the gap. It tests the ability of the legs to address a hazard in rapid succession.

- *Small object on plateau*: This terrain causes the robot to handle different types of challenges. It tests the robot's ability to coordinate obstacle avoidance which legs perform individually with hazard avoidance which the legs perform as a team.

8.3 Naturally occurring terrain

We tested Hannibal's ability to traverse naturally occurring terrain in two environments. The first environment is called the "sandbox". It's floor consists mostly of gravel, with some sand, and rocks of various sizes. Our lab built the sandbox to test Hannibal and our other robots in a more realistic setting. The second environment is Mars Hill in Death Valley, CA. This location has an uncanny resemblance to terrain images taken of Mars. It is characterized by a fairly firm, undulating, sandy surface with scattered rocks.

9 Results

Figure 17 quantifies Hannibal's performance at traversing various aspects of rough terrain. It shows the size range of obstacles that the robot can successfully negotiate. These results were gathered from the individual obstacle tests.

On average, Hannibal successfully traversed the sample terrains of the multiple obstacle tests, provided the terrain features remained within the values of figure 17. Similarly, Hannibal could also traverse natural terrain as long as the terrain was subject to similar constraints. If the obstacles exceeded these limits, Hannibal's feet and legs could get into awkward circumstances from which the robot could not recover. For example, Hannibal's feet could slip between gaps on the down-step, but get stuck in the crevice during the searching movements. Slippery feet were sometimes a problem when Hannibal tried to back away from ledges. Occasionally the front supporting foot could slip over the ledge while the robot tried to back away. Jagged rocks seemed to be difficult for the robot to walk over because it is difficult to obtain a secure foothold and relatively easy for a leg to get stuck on the rock's

protruding parts.

The rough terrain controller pushes the envelope of Hannibal's physical capabilities. Hannibal successfully handles a wide assortment of terrain features, but the maximum sizes of the obstacles are primarily determined by the physical capabilities of the robot. Obviously, if the robot had greater physical capabilities, such as larger work spaces for the legs, better leg kinematics, better sensing, or a greater strength to weight ratio, the robot could traverse more challenging terrain. These issues are beyond the scope of this work, and we will not address them further. It would be interesting to implement a similar control network on future legged robots for comparison.

10 Evaluation

Hannibal's rough terrain performance gives us a means to evaluate Hannibal's controller. In this section, we evaluate Hannibal s controller with respect to the controller design issues presented earlier in the paper, and the effectiveness of the local control with cooperation approach in achieving coherent global behavior.

10.1 Controller Design

Designing and implementing Hannibal's controller is an exercise in managing complexity. We found the subsumption approach implemented using the Behavior Language was effective for designing and implementing a controller for a robot of Hannibal's complexity.

- Scalability: The controller scales well for our application. Hannibal operates in real-time despite the large number (approximately 1500) of concurrently running processes.

It is impressive that Hannibal's full behavioral functionality is implemented on a single microprocessor and runs in real-time.

Since all of Hannibal's control code runs on a single processor, the controller code can be scaled until its requirements exceed what the master processor can provide. A logical step would be to implement behavior control on multiple processors. The robot's behavior control code is written so that it could be physically distributed provided that a processor is added to each subsystem. However, the Behavior Language would have to be modified to support multiple processors. The scalability of this alternate architecture is limited by the communication latency of the behavior code processors. This may not be a stumbling block for the current implementation since the subsystems mostly perform local functions and communicate with the other systems on a relatively limited basis.

- Modularity: In building a controller of this complexity from the bottom-up, we have demonstrated the modularity of the controller design. This modularity appears on every scale of the controller: Augmented finite state machines form agents, agents form task-achieving behaviors, task-achieving behaviors form levels of competence, levels of competence form a subsumption-based controller, subsumption-based controllers run within intelligent subsystems, and these subsystems cooperate to control the behavior of the overall robot.

  Over the course of this work, the robot and it's controller underwent many changes. Sensors were added or changed, actuated subsystems such as the body and legs were brought on-line, and new locomotion, sensor processing, and fault tolerance capabil-

ities were added to the system. The controller's modular organization allowed us to quickly add or change the processes which interpret new sensor information, utilize new actuators, or add new processing capabilities to the system.

- Flexibility: The controller's flexibility is demonstrated by the rough terrain tests. These tests require that the controller perform diverse tasks such as sense terrain hazards, perform basic mobility skills, and execute evasive maneuvers. The robot successfully completed each obstacle course provided the hazards remained within the limits of figure 17.

- Adaptability: The controller's adaptiveness is demonstrated by the rough terrain tests, which require that the robot's behavior readily adapt to changing terrain. Hannibal successfully negotiates small obstacles in its path, large obstacles blocking its path, gaps in the terrain, and cliffs.

10.2 Cooperating Local Controllers

We have demonstrated that local leg control with inter-leg communication can implement real-time control of a complex robot in an unstructured environment. Hannibal's implementation of locomotion capabilities separates local sensori-motor tasks from global communication tasks. In this way, each controller is responsible for managing relatively few sensors and actuators compared to the overall system. As a result, the load on the local controllers is relatively light. Thus, a tight coupling between sensing and action is maintained so that each local controller operates its leg in real-time. For example, in rough terrain locomotion, each leg is responsible for sensing and negotiating the rough terrain challenge it faces.

Consequently, all legs simultaneously handle local obstacles in real-time. This ability was demonstrated in the multiple small obstacles test.

The cooperation tasks are achieved through inter-leg communication. Two types of communication have been described: direct inhibition/excitation messages sent between controllers, or indirect messages distributed by broadcast agents. Through our examples, we have illustrated several ways this communication is used. First, each leg behaves as a scout for the other legs and alerts them of common dangers. In this manner, the local terrain view of each leg is shared with the other legs. This ability was demonstrated in the tests using cliffs and large obstacles. Second, inter-leg communication enables the legs to act as a team. With help from the other legs, each leg accomplishes more than it could on its own. For example, by asking the supporting legs to elevate the body, a leg can clear a taller obstacle. This was demonstrated in the individual obstacle tests using obstructive objects of varying heights and shapes. Third, inter-leg communication enables the legs to maintain a unified effort, and the inter-leg priority scheme maintained the unified effort even when behaviors of different legs were in conflict. This ability was demonstrated by the small plateau test.

11 Discussion

Why is Hannibal's controller so complex? Is it unnecessarily complex? It is understandable that more processes would be required to perform sensori-motor integration of a system with more sensors and actuators. Hannibal has considerably more sensors and actuators than a typical autonomous robot of today. It is also understandable that more processes would be required to add more behaviors to a robot's capabilities. Hannibal has a comparatively large

behavioral repertoire to that of a typical autonomous robot. It performs insect-like gaits, rough terrain locomotion, and possesses extensive failure recognition, fault tolerance, and sensor processing capabilities.

However, it is difficult to say what a reasonable requirement is on the number of processes necessary to implement a controller of comparable capabilities, and on a robot of similar physical sophistication, as Hannibal. To grapple this question, we compare the complexity of other insect-inspired controllers to the appropriate subset of Hannibal's controller. The four biologically motivated controllers presented in section 5 are used in the comparison. We refer to table 18 which summarizes the number of processes used by each controller to implement stable gaits on flat terrain and locomotion over small obstacles (where applicable). Different approaches were used, but we roughly equate one process, to one neuronal element, to one AFSM.

The complexity of Hannibal's controller compares rather favorably to the complexity of the other controllers . For flat terrain walking with one gait, Hannibal requires two fewer AFSMs than Genghis. For flat terrain walking with multiple gaits, Hannibal only requires one more process than that of the Case Western Hexapod. For locomotion over terrain with small obstacles, Hannibal requires the same number of AFSMs as Genghis. If one neuronal element is equated to one AFSM, Hannibal's controller is approximately two times *less* complex than that of Cruse. Treating this example as a guideline for the number of processes used to implement the entire controller, Hannibal's controller does not appear to be gratuitously complex.

12 Comparison of Insect-Inspired Controllers

Thus far we have argued that local control with cooperation is a viable control scheme for controlling a complex robot. We have used the example of rough terrain locomotion to make our case. As presented earlier, several other groups have used insect-inspired schemes to program their legged robots, or have explored these schemes in simulation. In this section, we compare the performance of our robot to those systems which have also employed biologically motivated mechanisms. Ferrell (1993) implements several biologically inspired controllers on the *same* robot, which provides an interesting comparison.

The controllers described in Donner (1987), Cruse (1994), and Beer & Chiel (1993) are more reflexive than the controller implemented on Hannibal in that all step cycle phase transitions are largely determined by inhibition/excitation signals sent between adjacent legs. In contrast, Hannibal's transitions are determined by the phase of local leg oscillators where the phase can be affected by excitation/inhibition signals sent from rough terrain behaviors. The appeal of these more reflexive locomotion control schemes is their elegance. It has been demonstrated either in simulation or on robots that these sorts of controllers provide a lot of functionality for a handful of clever mechanisms. Several interesting behaviors emerge from these controllers. For example, each controller has the ability to produce a family of stable insect-like gaits, the controller of Chiel et al. (1992) has demonstrated robustness to lesions, and the simulated controller presented in Cruse (1994) is resilient to leg perturbations caused by small obstacles.

However, these schemes have drawbacks as well. First, the gait coordination mechanism parameters require a fair amount of adjustment to achieve acceptable performance.

Cruse (1994) adjusted his simulation parameters based on results from studying stick insects. Other researchers have arrived at the proper weighting of influences through optimization techniques (Beer & Gallagher 1992), or by experimentation (Quinn & Espenschied 1993), (Ferrell 1993). Another drawback is the difficulty in adding new behaviors. The controller mechanisms interact in a very specialized way to produce gait coordination. Consequently it is difficult to incorporate dramatically different behaviors without disrupting controller dynamics. For example, locusts use rapid searching movements with a leg to locate a secure foothold (Pearson & Franklin 1984). This behavior is characterized by rapid, repeated cycling of a single leg while the other legs may even come to rest. It is not clear how to modify the current mechanisms to incorporate such an asymmetrical behavior. A typical approach is to simply add some sort of centralized process on top of the locomotion network that can override it (Donner 1987), but this detracts from the elegance of the reflexive approach.

Finally, there is a long way to go before robots using reflexive control schemes can locomote like insects over naturally occurring terrain. Typical mechanisms don't account for a lot of rough terrain locomotion behaviors observed in insects. Pearson found that locusts use several single leg tactics such as rhythmic searching movements when a leg does not contact a surface at the end of its swing, an elevator reflex to lift a leg above an object contacted during the swing phase, and local searching movements once a leg contacts a potential supporting surface. Pearson also found that animals do not adopt a rigid gait when walking on rough terrain. In fact, he observed a wide range of stepping patterns in which the gait pauses and shifts frequently. Progress is being made in this area, as evidenced by Cruse (1994), but much work has do be done.

In contrast, the appeal of the approach used on Hannibal is the ease of adding new

features to the system. On the downside, this makes for a bulkier controller – more code must be added which takes up more computing cycles. But the designer also has more control over the interactions of processes within the controller. This makes the controller easier to design around and build upon, although traditional issues like behavior conflict must be carefully resolved. (This is typically done using inhibition and suppression mechanisms, and testing the result through trial and error during the design phase.) For example, a wide assortment of rough terrain behaviors and terrain sensing processes were integrated into our basic locomotion controller. Several of these behaviors are inspired after those observed in insects such as local searching movements, an elevator reflex, stepping over holes, walking around large obstacles, and walking away from ledges. More reflexive insect-inspired locomotion schemes do not exhibit a comparable repertoire of rough terrain behaviors.

13 Conclusions

We demonstrated that the subsumption approach is effective for controlling robots of Hannibal's complexity. The complete implementation consists of approximately 1500 concurrent processes performing diverse tasks, such as terrain sensing, rough terrain locomotion, and fault tolerance. Through building a controller of this complexity for a robot with many sensors and actuators, we demonstrated the importance of controller scalability, modularity, flexibility, and adaptability. Hence, this work provides a point of reference for the ease of design and effectiveness of the behavior based approach in controlling more complex robots.

We showed that the *local control with cooperation* paradigm is an effective means of controlling a robot of Hannibal's complexity, and tested this approach by presenting Hannibal

with various test terrains. In regard to basic locomotion, communication between coupling agents causes various insect-like gaits to emerge. For rough terrain locomotion, inter-leg communication enables each leg to behave as a scout for the other legs and alert them of common dangers. In this manner, a high level terrain view local to each leg is shared with the other legs. Communication between legs also enabled them to act as a team, which enables each leg to accomplish more than it could alone. Finally, inter-leg communication enables the legs to maintain a unified effort, and the inter-leg priority scheme maintained the unified effort even when behaviors of different legs were in conflict.

This work of particular interest because the controller was implemented and tested on a physical robot of significant complexity. It is important to study behavior control systems on real robots. As robots become more complex, the difficult issues in control are magnified to the point where they cannot be ignored or only partially addressed as is often the case with simulation. It is important that all aspects of the control system be implemented and tested on real robots of sufficient complexity to make sure the *integrated* system works in the real world. By developing our controller on a physical system, we were forced to make sure all the pieces fit together at every stage. This also helped us maintain a common framework when implementing diverse capabilities. Finally, testing our controller on a physical system served as a strong validation test for our gait generation and rough terrain locomotion implementations.

14 Acknowledgments

This work was conducted under the supervision of Professor Rodney Brooks at the MIT Artificial Intelligence Laboratory. Special thanks to Rod Brooks, Mike Binnard for reviewing earlier versions of this work and providing useful comments.

References

Angle, C. (1991), Design of an Artificial Creature, Master's thesis, MIT.

Angle, C. & Brooks, R. (1990), Small Planetary Rovers, *in* 'Proceedings of IEEE International Workshop on Intelligent Robots and Systems', Ibaraki, Japan, pp. 383–388.

Bares, J. & Whittaker, W. (1990), Walking Robot with a Circulating Gait, *in* 'Proceedings of IEEE International Workshop on Intelligent Robots and Systems', Ibaraki, Japan, pp. 809–815.

Beer, R. & Chiel, H. (1993), Simulations of Cockroach Locomotion and Escape, *in* 'Biological Neural Networks in Invertebrate Neuroethology and Robotics', Academic Press, Inc., Boston, pp. 267–286.

Beer, R. & Gallagher, J. (1992), 'Evolving dynamical neural networks for adaptive behavior', *Adaptive Behavior* **1**, 91–122.

Brooks, R. (1986), 'A Robust Layered Control System for a Mobile Robot', *IEEE Journal of Robotics and Automation* **RA-2**, 14–23.

Brooks, R. (1989), 'A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network', *Neural Computation* **1:2**, 365–382.

Brooks, R. (1990), The Behavior Language; User's Guide, Technical Report 1227, MIT AI Lab.

Chiel, H., Quinn, R., Espenschied, K., & Beer, R. (1992), 'Robustness of a Distributed Neural Network Controller for Locomotion in a Hexapod Robot', *IEEE Transactions on robotics and automation* **8**, 293–303.

Connell, J. (1989), A Colony Architecture for an Artificial Creature, Technical Report 1151, MIT AI Lab.

Cruse, H. (1980), 'A Quantitative Model of Walking Incorporating Central and Peripheral Influences: I. The Control of the Individual Leg', *Biological Cybernetics* **37**, 131–136.

Cruse, H. (1990), 'Coordination of Leg Movement in Walking Animals', *European Conference on Artificial Life* **13**, 105–119.

Cruse, H. (1994), A Neural Net Controller for a Six-Legged Walking System, *in* 'Proceedings of From Perception to Action', Lausanne, Switzerland, pp. 55–65.

Donner, M. (1987), *Real Time Control of Walking*, Birkauser, Boston.

Ferrell, C. (1993), Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators, Technical Report 1443, MIT AI Lab.

Ferrell, C. (1994), 'Failure Recognition and Fault Tolerance of an Autonomous Robot', *Adaptive Behavior* **2**, 375–398.

Flynn, A., Brooks, R., Wells, W. & Barrett, D. (1989), Squirt: The Prototypical Mobile Robot for Autonomous Graduate Students, Technical Report 1220, MIT AI Lab.

Hirose, S. (1984), 'A Study of Design and Control of a Quadruped Walking Vehicle', *International Journal of Robotics Research* **3**, 113–133.

Maes, P. (1990), How to Do the Right Thing, Technical Report 1180, MIT AI Lab.

Mataric, M. (1990), A Distributed Model for Mobile Robot Environment-Learning and Navigation, Technical Report 1228, MIT AI Lab.

Payton, D., Keirsey, D., Kimple, D., Krozel, J. & Rosenblatt, K. (1992), 'Do Whatever Works: A Robust Approach to Fault-Tolerant Autonomous Control', *Journal of Applied Intelligence* **2**, 225–250.

Pearson, K. (1976), 'The Control of Walking', *Scientific American* **235**, 72–86.

Pearson, K. & Franklin, R. (1984), 'Characteristics of Leg Movements and Patterns of Coordination in Locusts Walking on Rough Terrain', *International Journal of Robotics Research* **3**, 101–112.

Quinn, R. & Espenschied, K. (1993), Control of a Hexapod Robot Using a Biologically Inspired Neural Network, *in* 'Biological Neural Networks in Invertebrate Neuroethology and Robotics', Academic Press, Inc., Boston, pp. 267–286.

Rosenblatt, K. & Payton, D. (1989), A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control, *in* 'International Joint Conference on Neural Networks', Washington, D. C., pp. 317–323.

Song, S. & Waldron, K. (1989), *Machnies that Walk, The Adaptive Suspension Vehicle*, The MIT Press, Cambridge, MA.

Stuart, W. K. (1988), Multisensor Modeling Underwater with Uncertain Information, Technical Report 1143, MIT AI Lab.

Wilson, D. (1966), 'Insect Walking', *Annual Review of Entomology*.

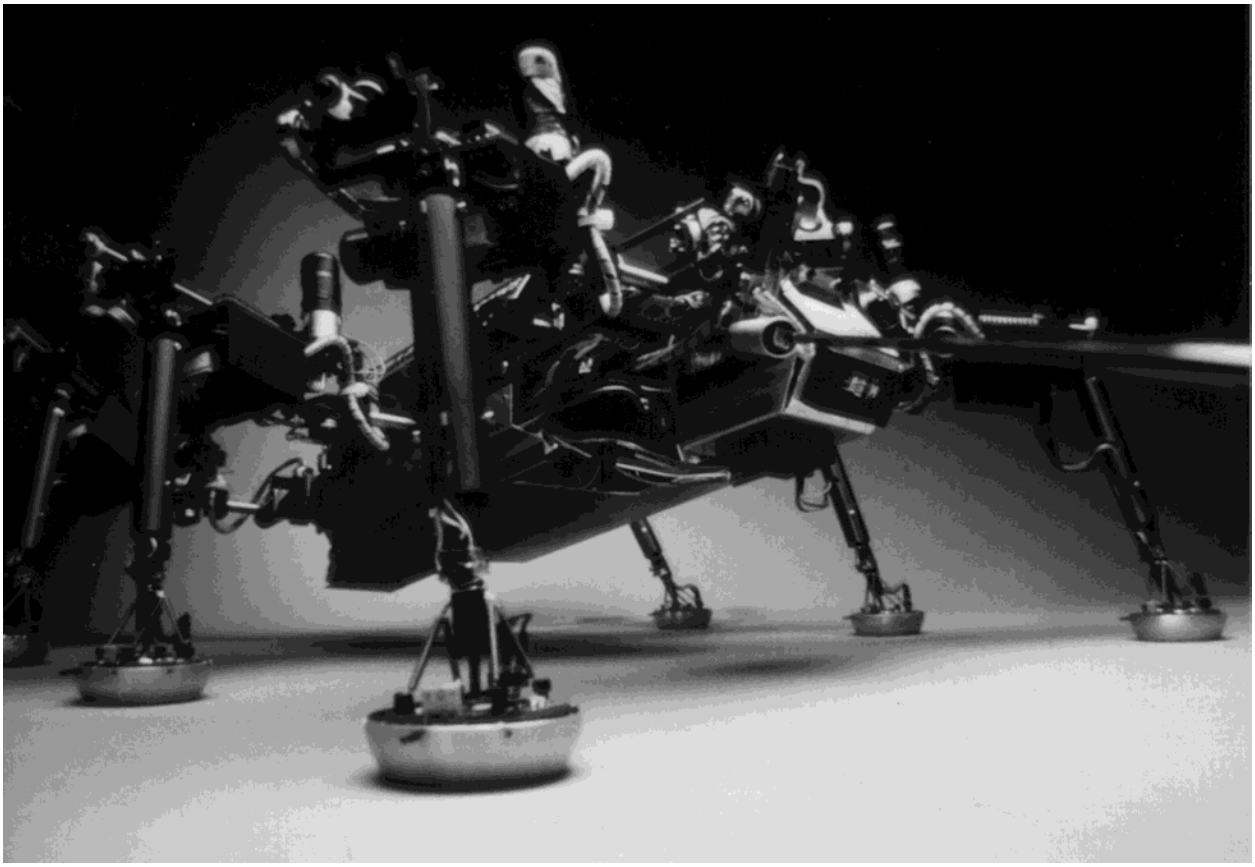

Figure 1: Hannibal is quite complex for its size. It is approximately the size of a bread box and is equipped with 19 degrees of freedom, over 60 sensors, and 8 computers.

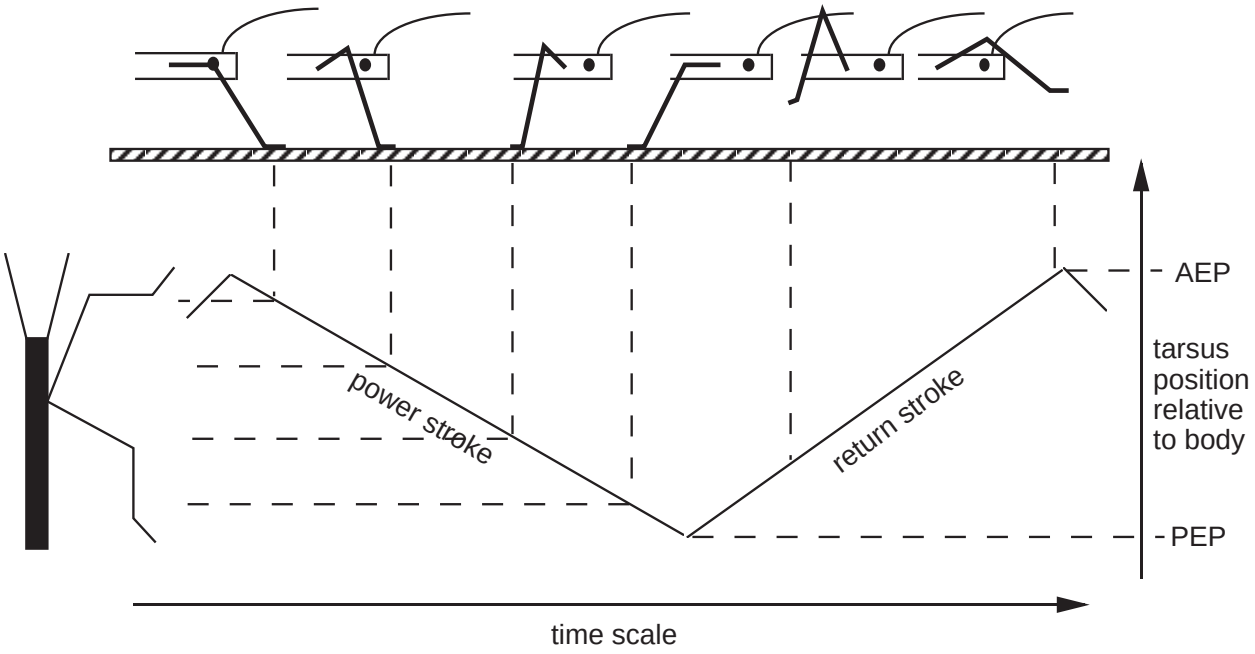

Figure 2: During the return stroke, the leg lifts and swings to the start position of the next power stroke. In forward locomotion, the leg moves towards the AEP during the recovery phase. During the power stroke, the leg supports and propels the body along the direction of motion. In forward walking, the leg moves toward the PEP during the support phase. Adapted from (Cruse 1990).

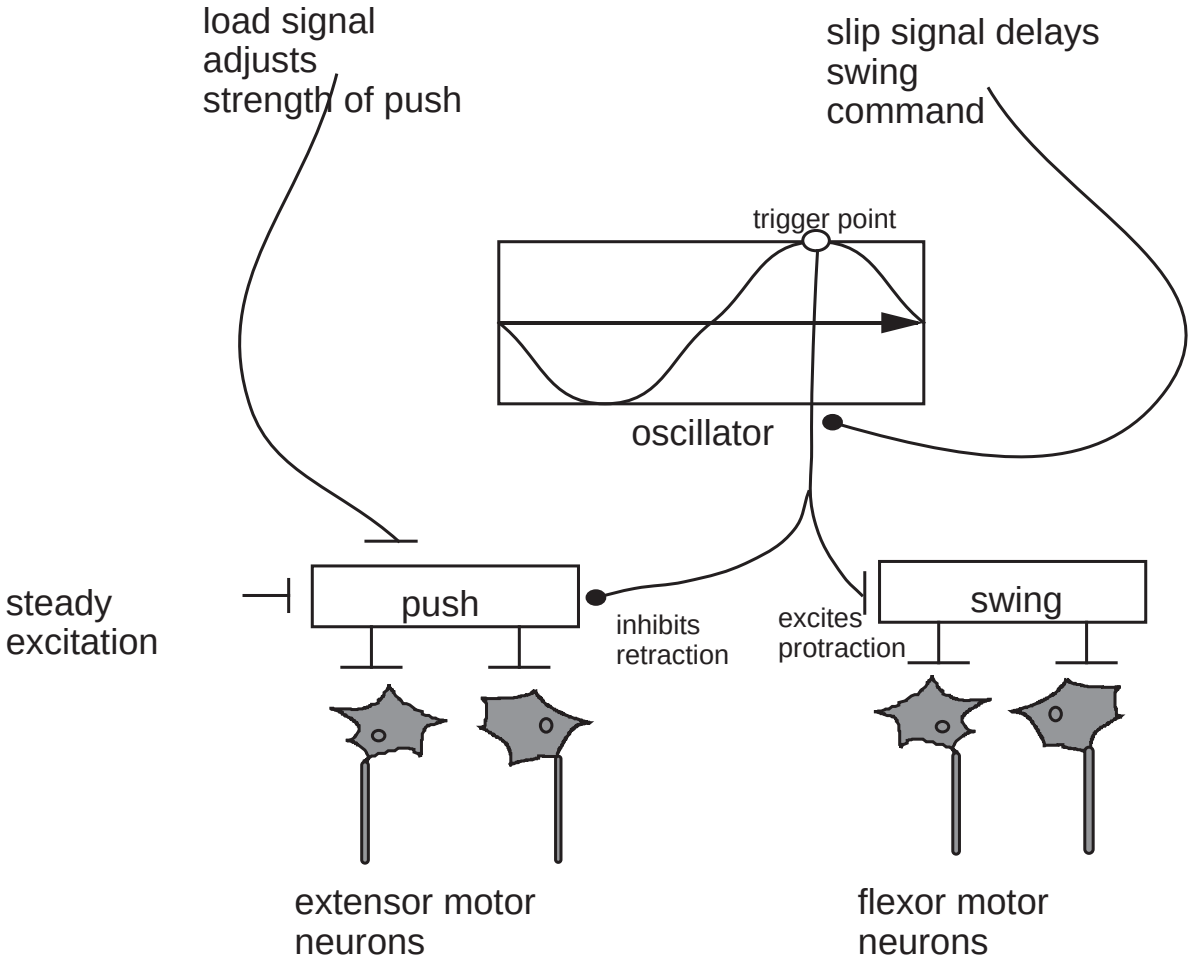

Figure 3: Pearson's neural circuit for controlling the stepping motion of a single leg. An oscillator provides the stepping rhythm. It triggers a swing command near the peak of its cycle. The swing command excites the motor-neuron circuit that swings the leg forward, and inhibits the push circuit. The push circuit presses the foot to the ground and draws it back. A steady excitatory input keeps the push circuit active whenever it is not inhibited by the swing command. Adapted from (Wilson 1966).

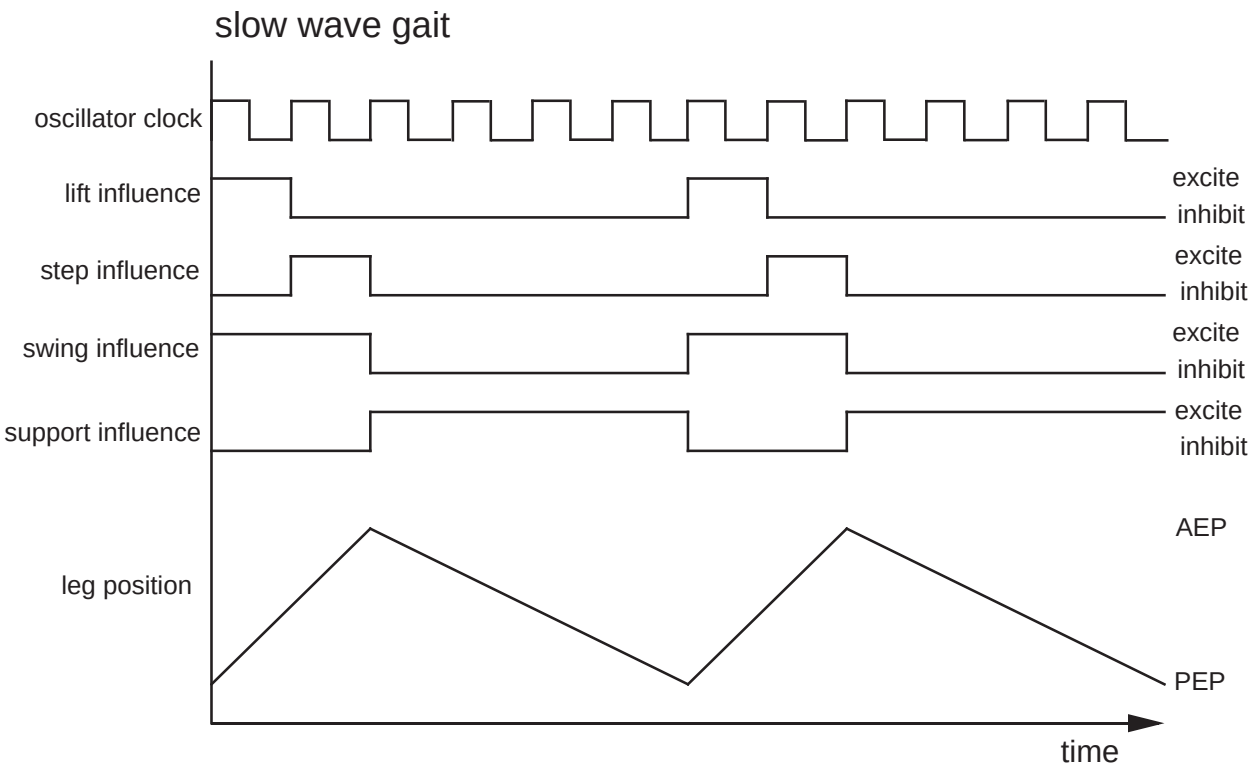

Figure 4: The influences used in the step cycle generation mechanism. The leg lifts and swings for the first clock stage, steps and swings for the second clock stage, and supports and propels the body for the remainder of the oscillator period.

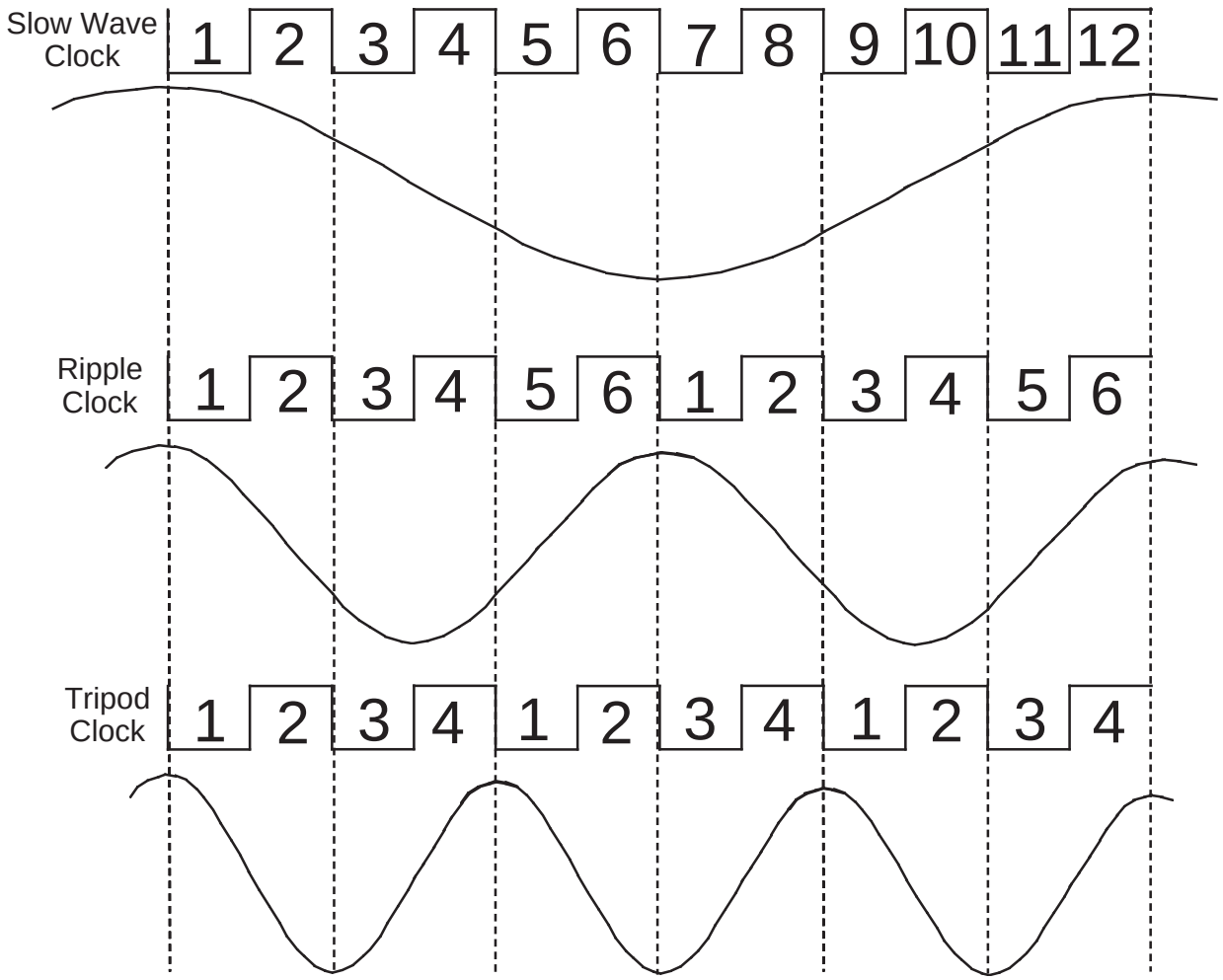

Figure 5: Implementation of leg oscillators on Hannibal. Each oscillator is modeled as a clock which cycles through its values at regular time intervals. The peak of the oscillator phase corresponds to a clock value of "1". The period of the clock corresponds to the period of the oscillator, which changes for the different gaits.

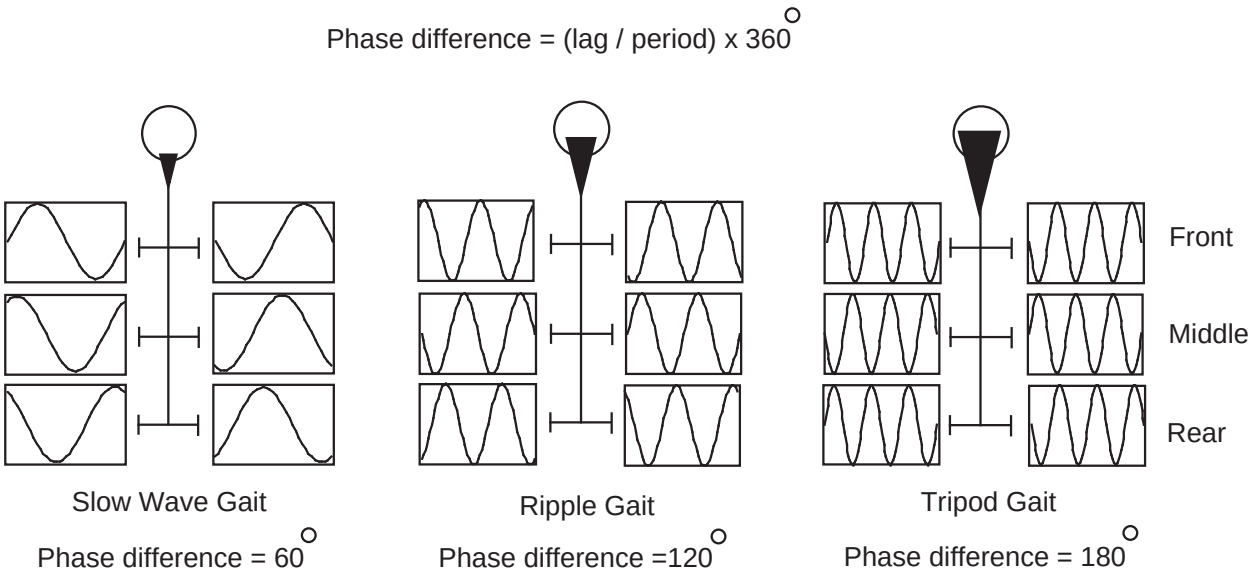

Figure 6: Various gaits emerge from changing the frequency of the local leg oscillators. Each gait has a characteristic phase difference between the ipsilateral legs. Adapted from (Wilson 1966).

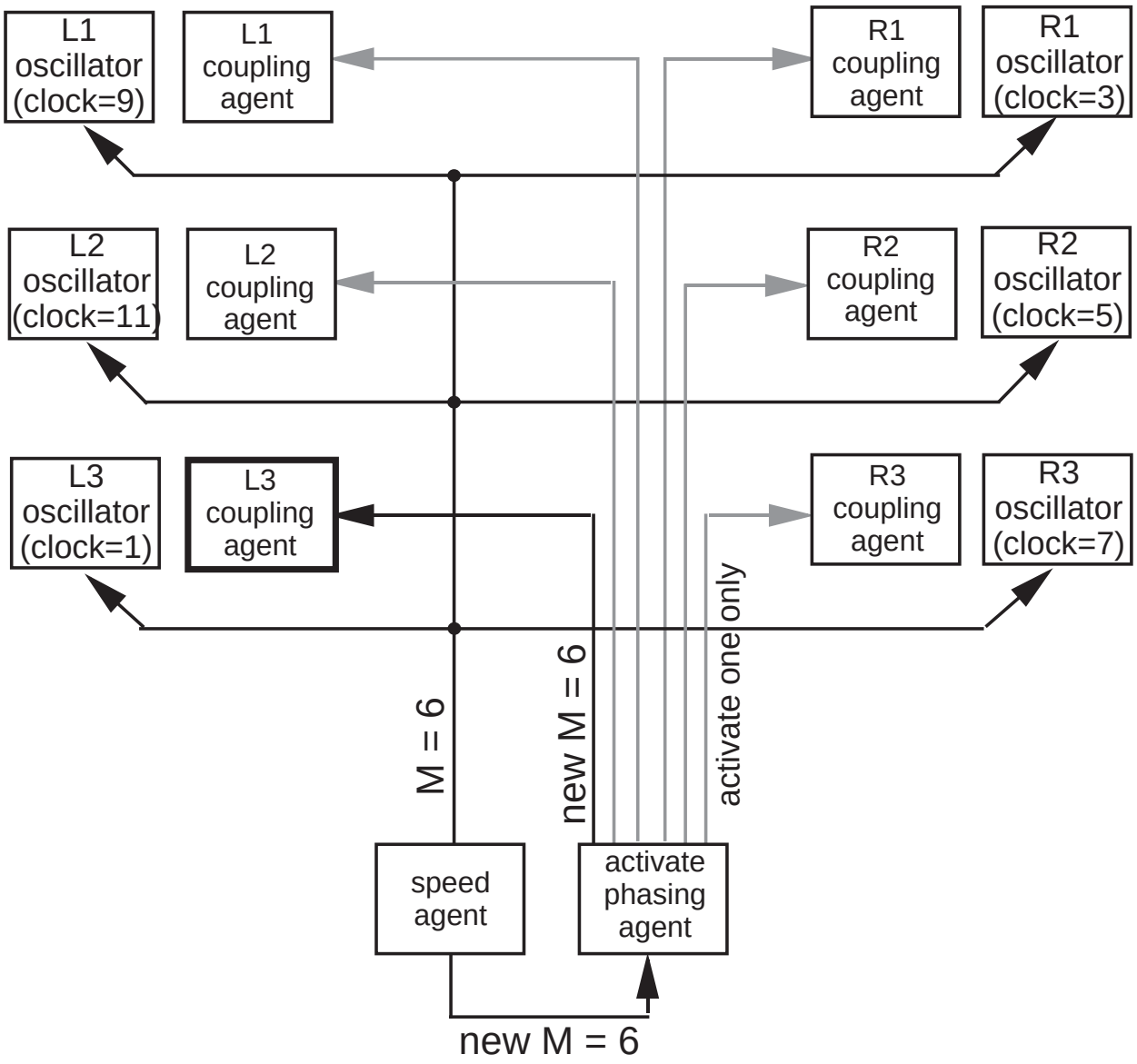

Figure 7: Circuit for activating a coupling agent. Initially, M=12 which evokes a slow wave gait. The Speed agent sends a new M value to all the leg oscillators to change the oscillator frequency. In this example, it sends the message M=6, which evokes a ripple gait. The Activate Phasing agent becomes active when it receives a new M value from the Speed agent. Once activated, it determines the most posterior leg in the lift-to-swing stage (clock=1), and activates its coupling agent. In this example, the left rear leg's coupling agent is activated. The Activate Phasing agent also passes along the new M value to this coupling agent.

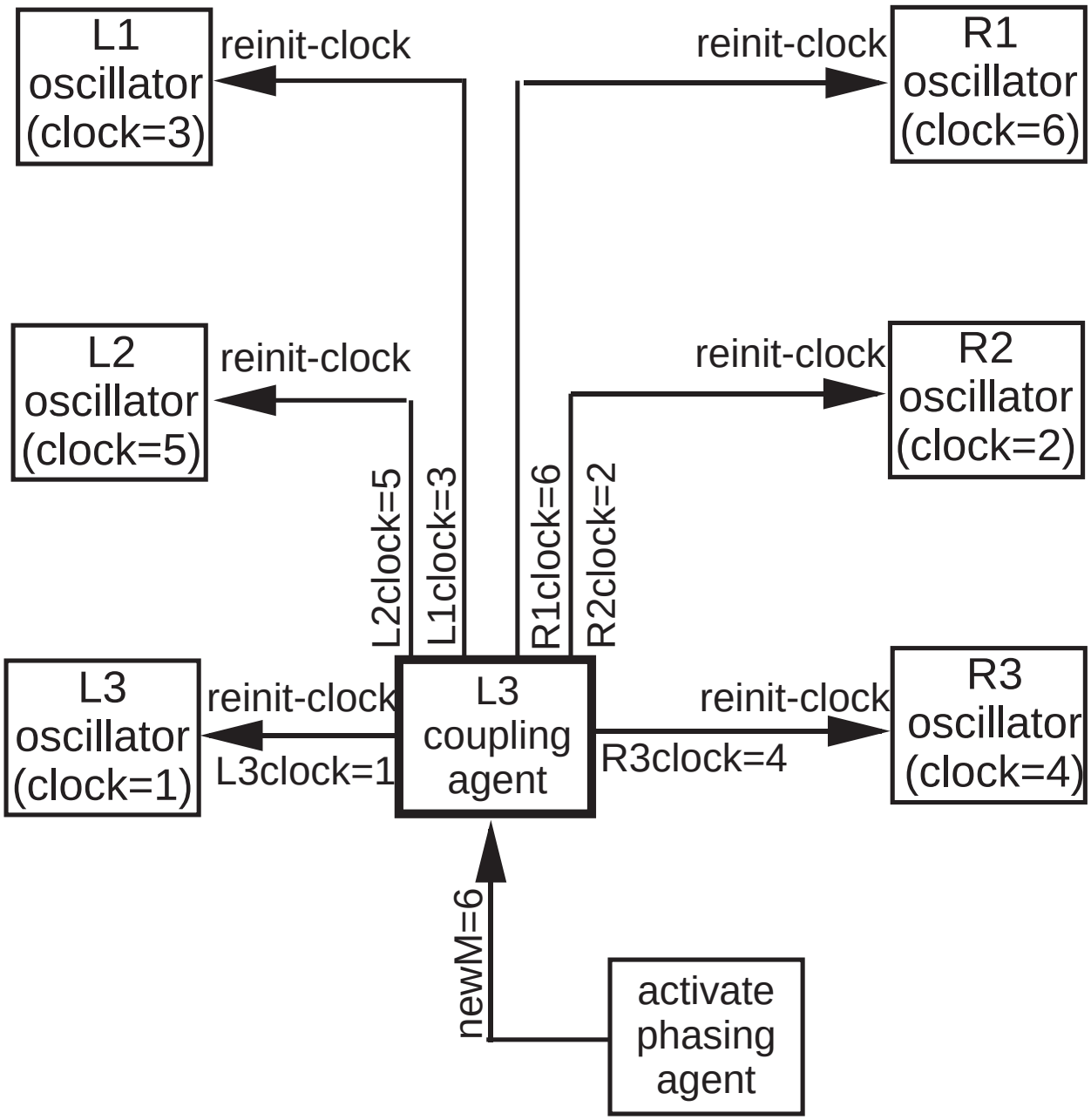

Figure 8: Circuit for re-phasing leg oscillators. Once activated, the coupling agent re-phases the leg oscillators by re-initializing the oscillator clock values of the legs. It uses the new M value (passed to it by the Activate Coupling agent) to determine the proper re-initialization values. It sets the oscillator clock values of the other legs with respect to the oscillator clock value of its leg.

```
(defbehavior L3-coupling-agent
  :inputs (newM)      ;; incoming message  = no. stages/osc. period
                      ;; incoming received only when command speed changes.
  :outputs (L3clock  ;; outgoing messages = reinit. osc. clock values
            L2clock
            L1clock
            R3clock
            R2clock
            R1clock)
  :activation (+ received-activation 10)
  :received-activation-range (-30 30)
  :threshold 10
  :continuance 0.05
  :i-processes ((whenever (received? M)
                  (cond ((= M $slow_wave_gait)
                         (output L3clock 1)
                         (output L2clock 11)
                         (output L1clock 9)
                         (output R3clock 7)
                         (output R2clock 5)
                         (output R1clock 3))
                        ((= M $ripple_gait)
                         (output L3clock 1)
                         (output L2clock 5)
                         (output L1clock 3)
                         (output R3clock 4)
                         (output R2clock 2)
                         (output R1clock 6))
                        ((= M $tripod_gait)
                         (output L3clock 1)
                         (output L2clock 3)
                         (output L1clock 1)
                         (output R3clock 3)
                         (output R2clock 1)
                         (output R1clock 3))))
            ))
;; make connections from coupling agent to leg oscillators
(connect (L3-coupling-agent L3clock) (L3-osc reinit-clock))
(connect (L3-coupling-agent L2clock) (L2-osc reinit-clock))
(connect (L3-coupling-agent L1clock) (L1-osc reinit-clock))
(connect (L3-coupling-agent R3clock) (R3-osc reinit-clock))
(connect (L3-coupling-agent R2clock) (R2-osc reinit-clock))
(connect (L3-coupling-agent R1clock) (R1-osc reinit-clock))
```

Figure 9: Code segment for the coupling agent of the rear left leg. This code segment is illustrated by figure 8.

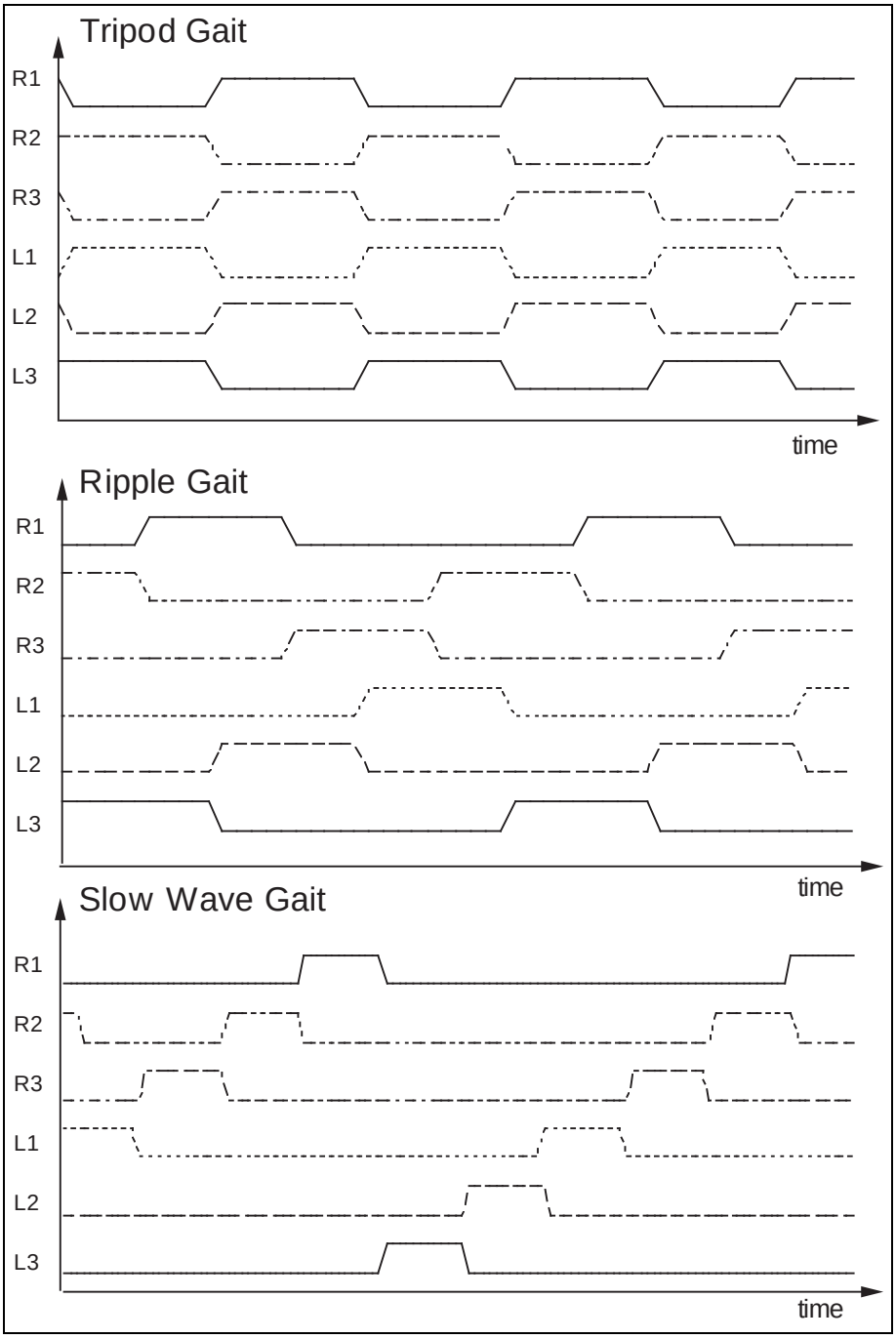

Figure 10: Run-time data of various gaits implemented on Hannibal. R1, R2, and R3 correspond to the right front, right middle, and right rear legs respectively, and L1, L2, and L3 correspond to the left front, left middle, and left rear legs respectively. Each wave form is high when the recovery phase excited, and low when the support phase is excited.

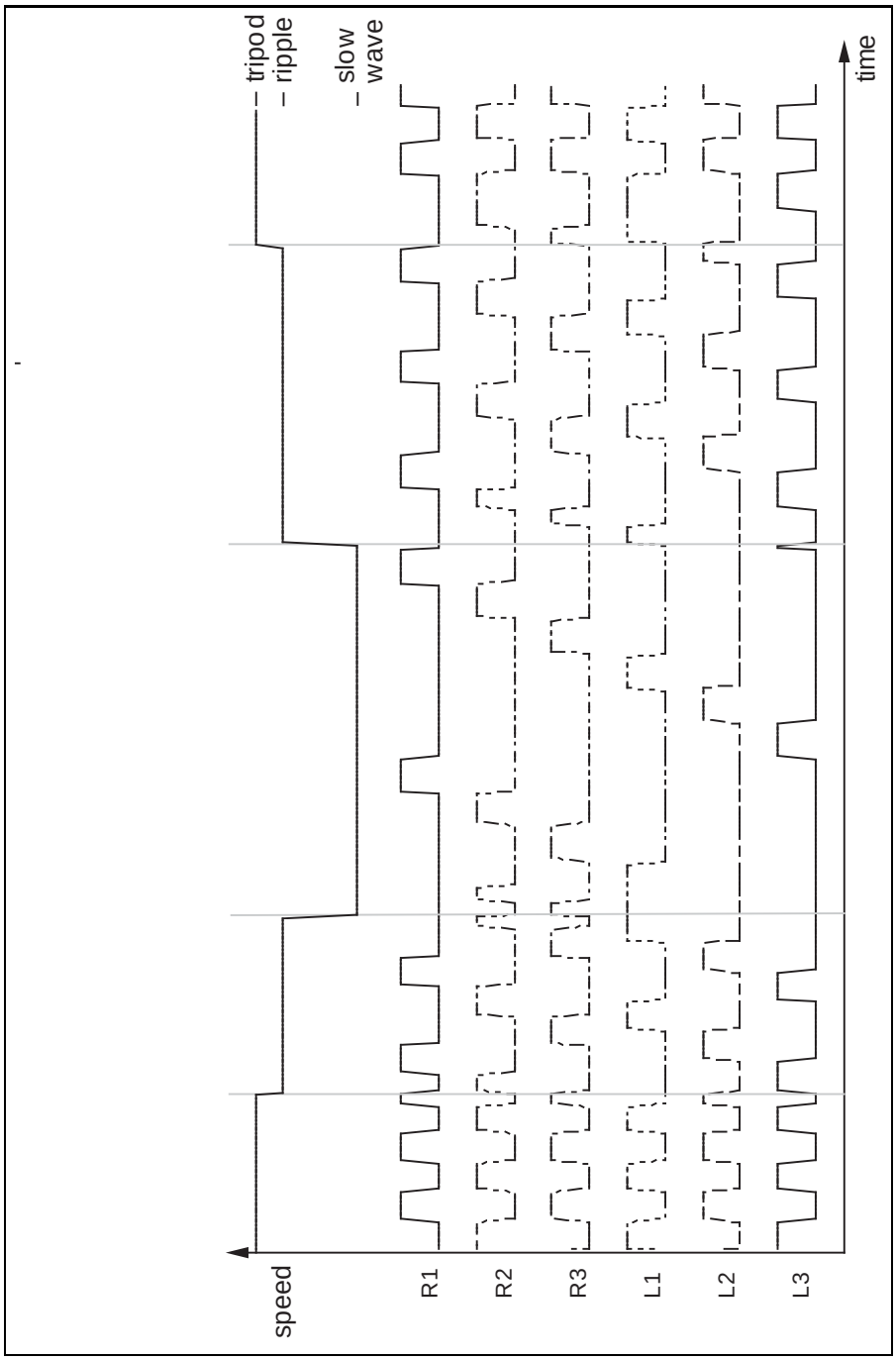

Figure 11: Run-time data of the robot switching between gaits as the speed command changes the oscillator frequencies. Each wave form is high when the recovery phase excited, and low when the support phase is excited. When the speed command changes, some of the legs become out of phase. As the leg oscillators re-synchronize themselves, they modify the recovery phase of these legs. As a result, the duration of the recovery phase excitations varies as the speed command changes. The observed gait transitions are stable and quick.

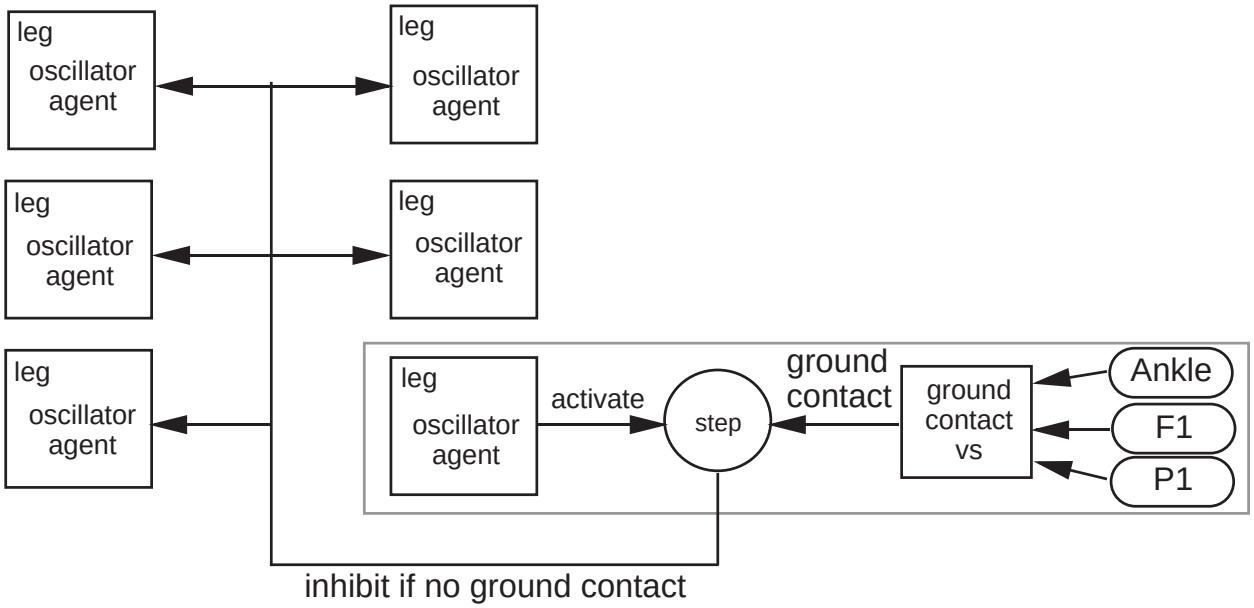

Figure 12: To enforce stable locomotion, the step agents of the recovering legs prevent the robot from advancing to the next step cycle unless the recovering legs support the body. Here, the right rear leg is in the recover phase. The step agent of this leg inhibits the oscillators of the supporting legs until the output of ground-contact virtual sensor of this leg is true. When this is the case, the step agent releases the oscillators of the other legs. Ankle, F1, and P1 are the physical sensors that measure ankle displacement, vertical force, and vertical position respectively.

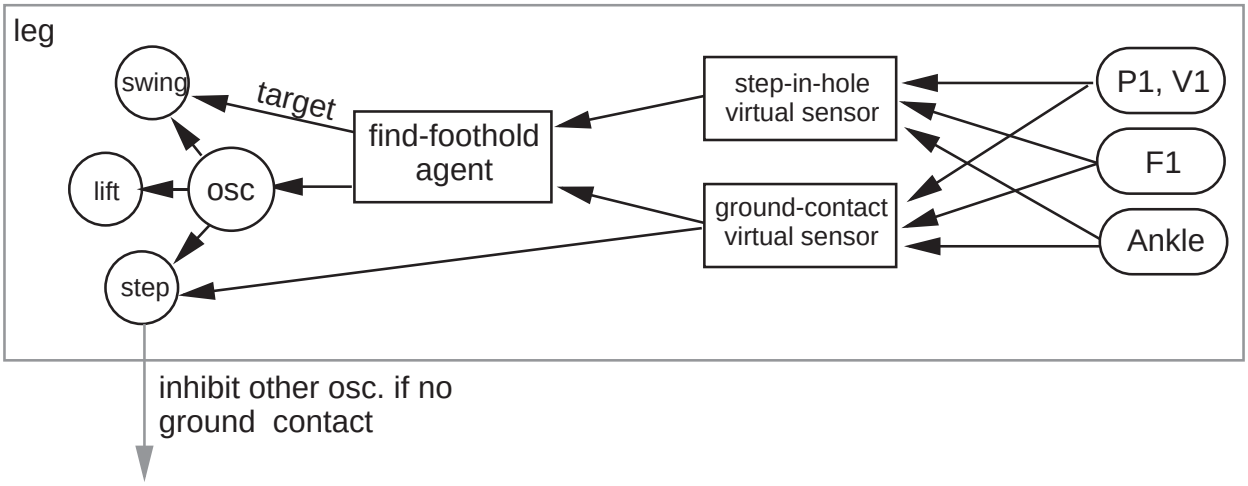

Figure 13: For each leg, the find-foothold agent is activated once the output of the step-in-hole virtual sensor is true. While active, it generates the searching pattern by making the leg's oscillator agent repeat the recover phase with new target values. Meanwhile, the leg's step agent makes the robot pause by inhibiting the oscillators of the supporting legs. The find-foothold agent is de-activated when the output of the ground-contact virtual sensor is true. Ankle, F1, P1, and V1 are the physical sensors that measure ankle displacement, vertical force, vertical position, and vertical velocity respectively.

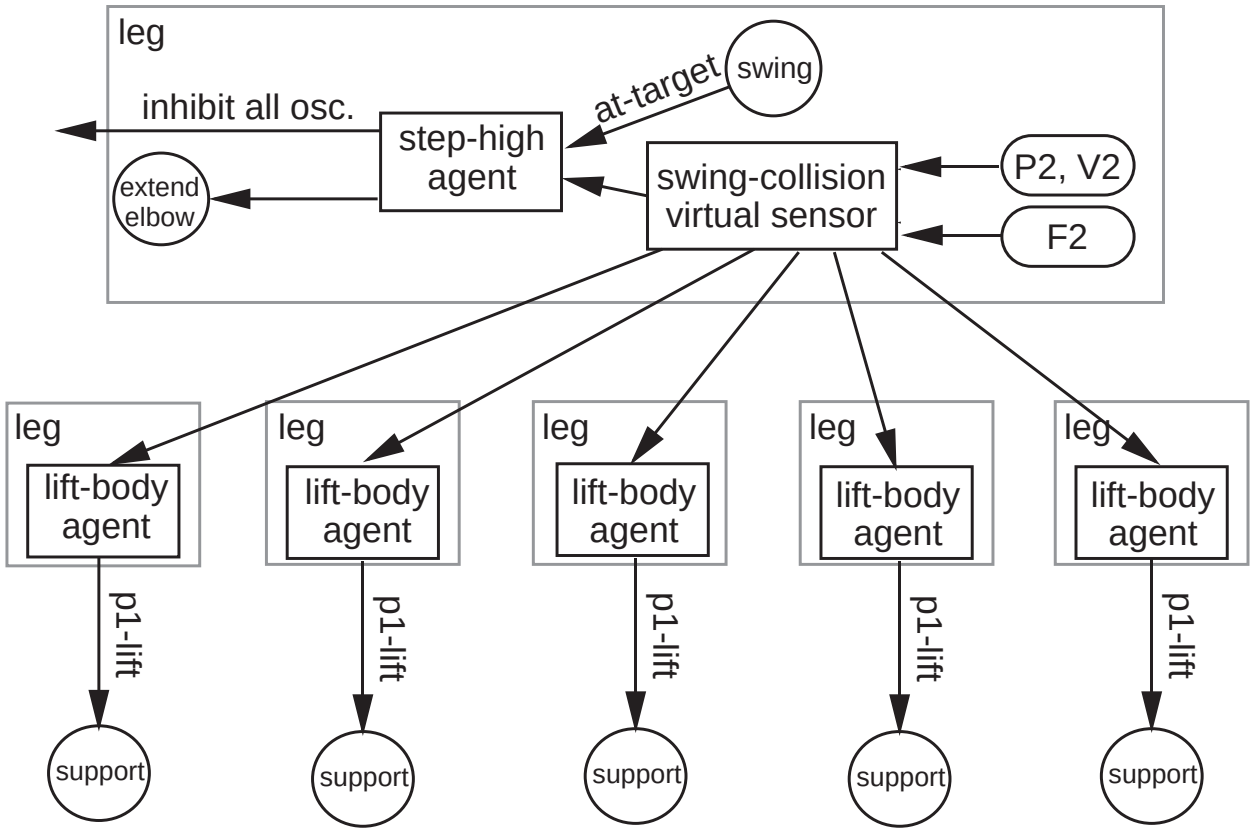

Figure 14: For each leg, the step-high agent is activated when the output of the swing-collision virtual sensor is true. When active, the step-high agent lifts the foot by activating the extend-elbow agent. Once the elbow is fully extended, the leg swings to its target value. Meanwhile, the lift body agents of the supporting legs are also activated while the swing-collision virtual sensor is true. This helps the collided leg clear the obstacle by elevating the body. The swing agent de-activates the step-high agent when the leg reaches the target value. Once this occurs, the elbow is contracted and the leg steps down.

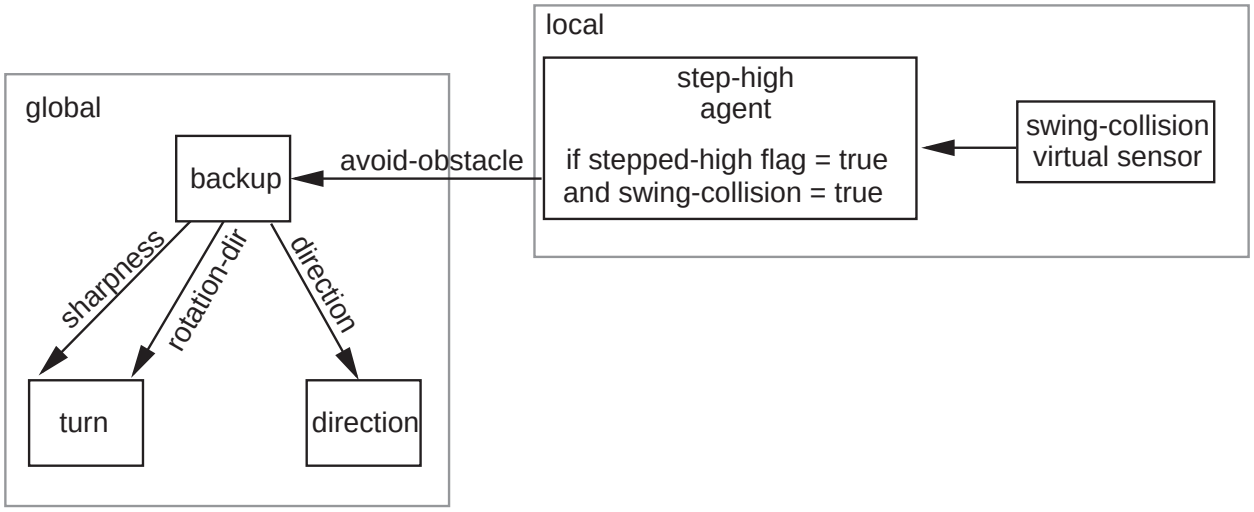

Figure 15: This network causes the robot to back away from obstacles that are too large to step over. The backup agent is activated when a leg collides with and obstacle after its foot is lifted as high as possible off the ground. The broadcast backup agent sets the number of steps the robot walks backwards and the turn sharpness based on the avoid-obstacle message. The direction of rotation is set such that the robot turns away from the obstacle.

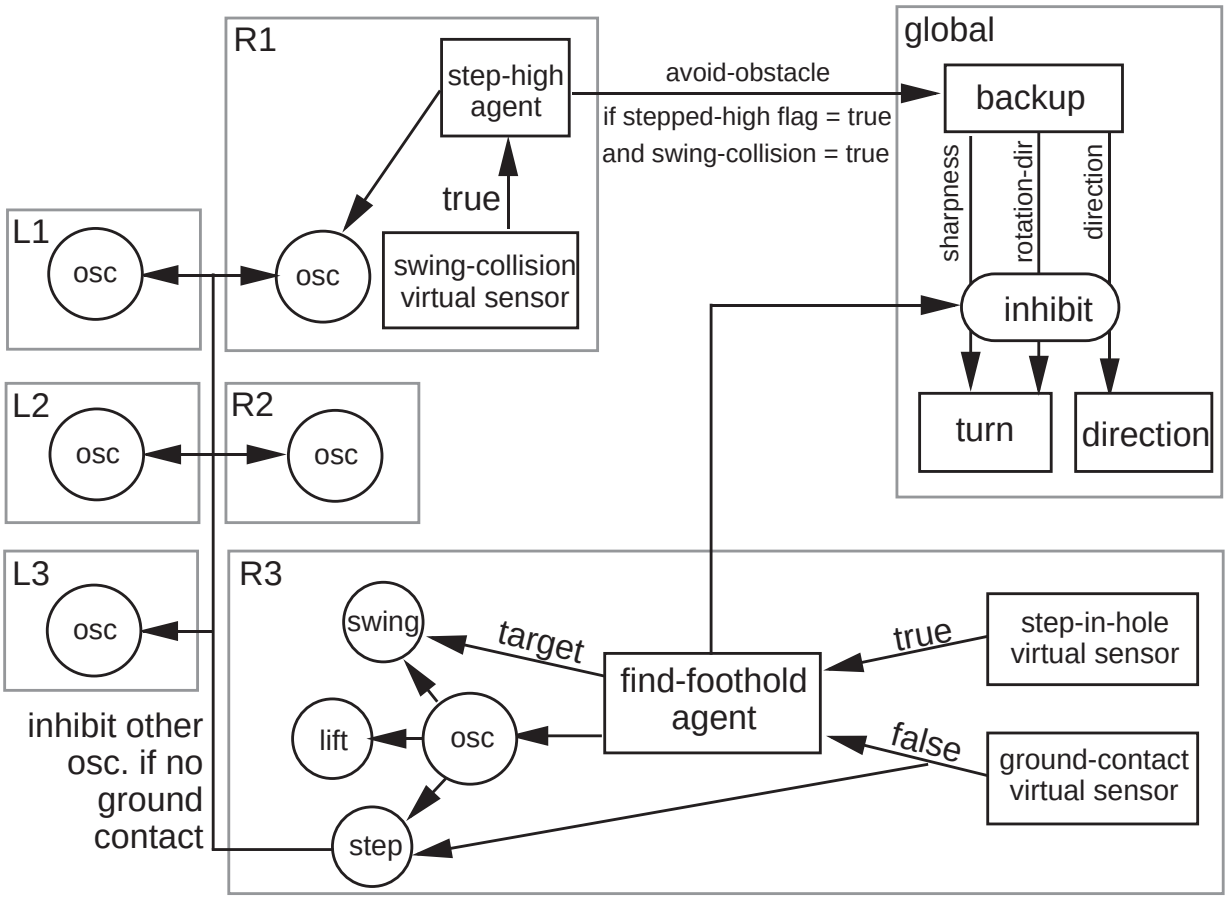

Figure 16: Inter-leg behavior conflicts are handled by a pre-programmed priority scheme. Behaviors that achieve goals with a higher priority inhibit behaviors that achieve goals with a lower priority. For example, stability has a higher priority than obstacle avoidance. As a result, the find-foothold behavior inhibits the backup behavior. Once the leg finds a foothold, the robot is allowed to backup.

terrain feature	traversable size		rough terrain behavior
	minimum size	maximum size	
small obstacle	1/4 average body height	1/3 average body height	step high over obstacle
medium obstacle	1/3 average body height	5/6 average body height	step high over obstacle and lift body
large obstacle	5/6 average body height	———	walk around
gap width	1 foot diameter	3/4 stride length width	search for foothold and step over gap
cliff depth	1/4 average body height	———	search for foothold, back up and turn away from cliff
incline slope	5 degree slope	15 degree slope	lean body forward
decline slope	5 degree slope	20 degree slope	lean body backward

Figure 17: Hannibal successfully traverses rough terrain as long as the obstacles remain within the limits of this table. The minimum size column lists the smallest obstacle that will evoke the corresponding behavior. The maximum size column lists the largest terrain feature the robot can negotiate with the corresponding behavior.

controller	number of processing elements
Cruse	120 neuronal elements (multiple gaits, small obstacles)
CW Hexapod	37 neurons (multiple gaits, flat terrain only)
SSA Hexpod	26 processes (single gait, flat terrain only)
Genghis	32 AFSMs (flat terrain, single gait) 56 AFSMs (single gait, low obstacles)
Hannibal	30 AFSMs (flat terrain, single gait) 38 AFSMs (flat terrain, multiple gaits) 56 AFSMs (multiple gaits, small obstacles)

Figure 18: Comparison of the number of elements used to implement several biologically motivated locomotion controllers. For this table, 1 neuronal element = 1 process = 1 AFSM. This is a very rough comparison since the amount of computation performed by each element varies by implementation. For comperable locomotion behaviors, the number of processes used to implement Hannibal's controller is similar to the number used by other controllers.