Goal-Seeking Behavior in a Connectionist Model

Thomas E. Portegys, Lucent Technologies, portegys@lucent.com

Abstract

Goal-seeking behavior in a connectionist model is demonstrated using the examples of foraging by a simulated ant and cooperative nest-building by a pair of simulated birds. The model, a control neural network, translates needs into responses. The purpose of this work is to produce life-like behavior with a goal-seeking artificial neural network. The foraging ant example illustrates the intermediation of neurons to guide the ant to a goal in a semi-predictable environment. In the nest-building example, both birds, executing gender-specific networks, exhibit social nesting and feeding behavior directed toward multiple goals.

1 Introduction

1.1 Purpose

The purpose of this work is to produce lifelike behavior with a goal-seeking artificial neural network. This is demonstrated using the examples of foraging by a simulated ant and cooperative nest-building by a pair of simulated birds. The foraging ant example illustrates the intermediation of neurons to guide the ant to a goal in a semi-predictable environment. In the nest-building example, both birds, executing gender-specific networks, exhibit social nesting and feeding behavior directed toward multiple goals.

1.2 Background

The natural world poses many challenges to animals for survival and reproduction which have fostered the evolution of capabilities beyond those of current machines. For example, computers may excel at monitoring the global stock market, but have difficulty harvesting fruit in an orchard. The human animal has also evolved intelligence, and according to James Albus (1979), plausibly from prior neural mechanisms:

> The rarity and late arrival of the ability to plan suggests that a highly developed precursor, or substrate was required from which planning capabilities evolved... The implication is that a sensory-interactive, goal-directed motor system is not simply an appendage to the intellect, but is rather the substrate in which intelligence evolved.

As a corollary, in his AAAI-98 Presidential Address, David Waltz (1999) offers this conjecture:

> ...What is intelligence for? That one I think we have some chance of answering. And the answer to it is that any kind of intelligent phenomenon that we see, whether physical or behavioral, is there because it really serves the organism's survivability purposes.

Reductionistic and somewhat behavioristic (Skinner 1957) views such as these suggest that intelligence emerges from basic goal-seeking brain mechanisms.

The goal of artificial intelligence is to create synthetic brains, and modeling neurological systems is a common course of action. But in what manner should machines model brains? At the fine-grained level of neurons? By simulating brain structures? This project takes the abstract, normative approach that "classic" artificial neural networks do. Steven Hampson (1990) offers this rationale for such abstractions:

> Connectionism, like artificial intelligence (AI), has no necessary commitment to biological relevance, but it is generally assumed that a better understanding of biological intelligence has something to offer to the study of artificial intelligence.

While artificial models may not be of direct relevance to biology (Dudai 1989), there is no conclusive evidence that the brain is either the only or the even best possible intelligent mechanism. To use an aeronautical analogy, the study of bird flight was useful, but not constraining, in the development of the airplane.

The model selected for this project, Mona, was introduced by Portegys (1999) as a connectionist model of motivation. A connectionist model bears a functional resemblance to a natural

neurological system in that many local interactions between units produce an emergent system effect. A type of control neural network (Fu 1994), Mona produces goal-seeking responses based on environmental input aimed at reducing homeostatic needs (Parten 1990). In living organisms these are inherent needs such as thirst, hunger, sex, etc. In the context of Mona, motivation, often a loosely defined term (Pfaff, 1982; McClelland 1987), is interpreted as a function which translates needs into responses.

2 The Mona Model

2. 1 Neurological Abstraction

Human and animal intelligence is performed by a network of neurons which operate by mutual excitation and inhibition (Thompson, Berger, and Berry, 1980; Holmes and Rall, 1992). Mona is an abstraction of a neurological system consisting of a network of computational units, each of which is capable of receiving and expressing mutually mediating influences. Consider a functional view of the nervous system of a simple organism which controls feeding behavior, shown in Figure 1. Feeding consists of the sequence of catching, killing and eating prey.

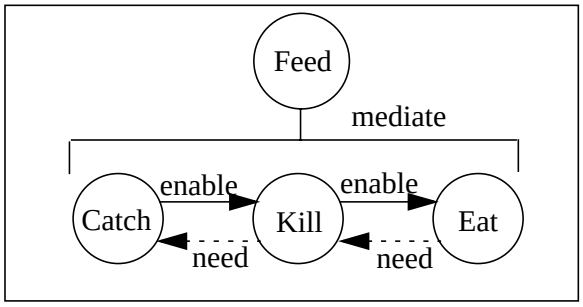

Figure 1 - Feeding control

“Feed”, “Catch”, “Kill”, and “Eat” are neurons which fire when their namesake events occur. The solid arrows are enabling (or disabling) signals directed from one neuron to another; these signals are analogous to the excitatory and inhibitory influences of living neurons. The dotted arrows are signals derived from the organism’s need for food. The above is read as follows. When the organism becomes hungry, the goal associated with the need of hunger-reduction, “Eat”, becomes a source of need signals propagating to antecedent neurons in a kind of “bucket brigade” from primary to secondary goals, causing them to become capable of responding. Once the prey is caught, the killing neuron is enabled, and once that is done, the eating neuron is enabled. The enabling of a neuron means that a context has been established in which it may successfully fire. A neuron’s state thus consists of the 3-tuple {need, enablement, firing}. The events with which neurons are associated can be drawn from sensors, responses, or, as in the case of the mediating “Feed” neuron, the states of component neurons.

Neurons maintain a base level of enablement, analogous to long term memory, which is dynamically modified, as part of short term memory, to accomplish a concerted operation. For example, the “Feed” and “Catch” neurons might be enabled by default, while the “Kill” and “Eat” neurons rest in a disabled state awaiting the catching of prey. This would prevent an attempt to kill an object being caught by the organism for the purpose of mating or building a nest.

The ability of neurons to disable other neurons allows further opportunity for context-dependent cooperation. For example, it would be sensible for the “Catch” neuron to disable itself upon firing to prevent the seizing of prey while a catch is being eaten.

2.2 Description

Mona has a simple interface with the environment, shown in Figure 2, similar to that utilized by Portegys (1986) and Nolfi and Parisi (1996). All knowledge of the state of the environment is absorbed through “senses”; there are no special modalities or channels by which instructions or meta-information are given. Responses are expressed to the environment with the goal of eliciting sensory inputs which are internally associated with the reduction of needs.

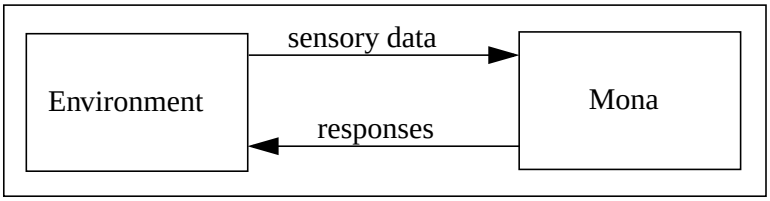

Figure 2 - The Mona/environment interface

The events which neurons represent can be drawn from sensors, responses, or the states of component neurons, calling for three types of neurons. Neurons attuned to sensors are “receptors”, those associated with responses are “motors”, and those mediating other neurons are “mediators”. A receptor neuron can process a logical combination of multiple sensor data. A mediator neuron controls the transmission of need and enablement through the sequence of its component neurons.

To elucidate by example, consider this task: Mona must get into her home from somewhere out in the world, a locked door barring the way inside, thus necessitating the use of a key to unlock the door. She needs to know several things, such as how to get to the door, how to unlock the door, and how to enter her home through the unlocked door. Mona must produce a sequence of responses to proceed from an initial keyless condition in the world to her home.

Figure 3 depicts the portion of Mona's neural network which manages the entering of home through an unlocked door. Let the house-shaped objects be receptor neurons, such as the one marked "Door"; the inverted houses be motor neurons, such as "Move"; and the diamonds be mediator neurons, such as "Enter home". The numbers in parentheses indicate need levels, which will be discussed presently; suffice it to say for now that the "Home" receptor has been associated with the reduction of a need, and is thus a goal for Mona. The numbered arrows proceeding from a mediator indicate a sequence of neurons mediated by it, known to the mediator as its "events". In this case, "Enter home" mediates a sequence of events associated with the receptor "Door", the motor "Move", and the receptor "Home". This mediator thus governs the process of entering home by moving through a door. The type of mediation exerted by "Enter home" is an enabling one, meaning that it allows firing events to propagate enabling influences. Although not depicted in this example, a disabling mediator has dotted arrows instead of solid.

Initially the door is locked, thus the "Enter home" mediator is disabled, meaning that it cannot function until preconditions establish an enabling context for it. This is represented by the dotted outline of the mediator. In order to enable "Enter home", another mediator must come into play: "Enable enter home". This mediator will enable the "Enter home" neuron when the "Unlock door" neuron fires.

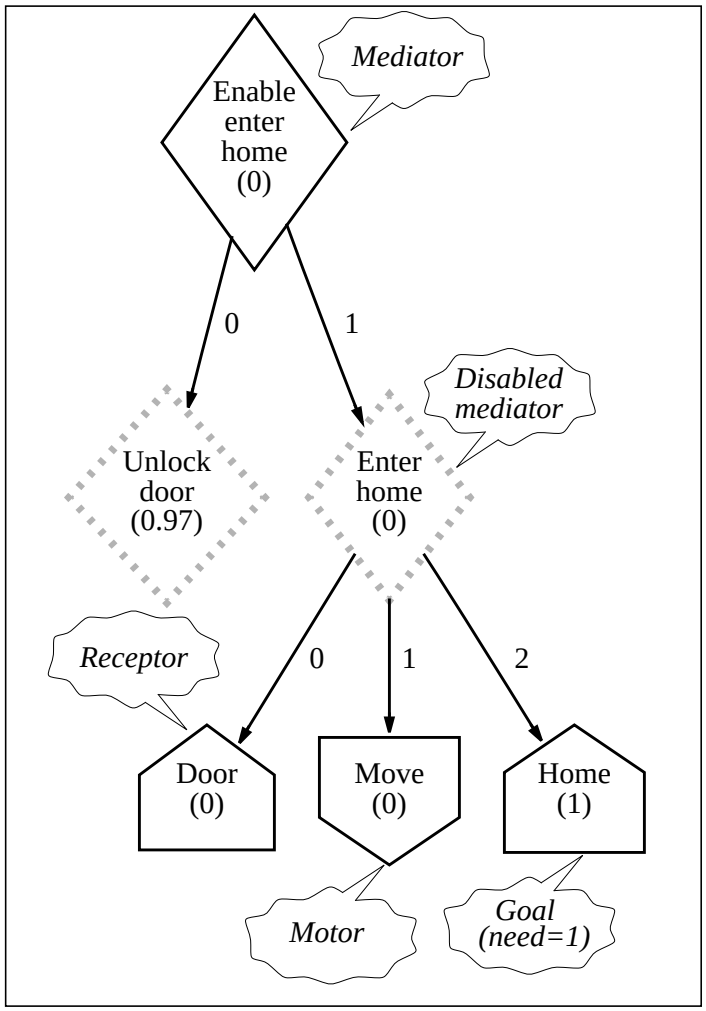

Figure 3 - Enable enter home/Enter home

However, the “Unlock door” neuron is also in a disabled state, requiring “Get key”, shown in Figure 4, to fire as a precondition - the door cannot be unlocked without the key.

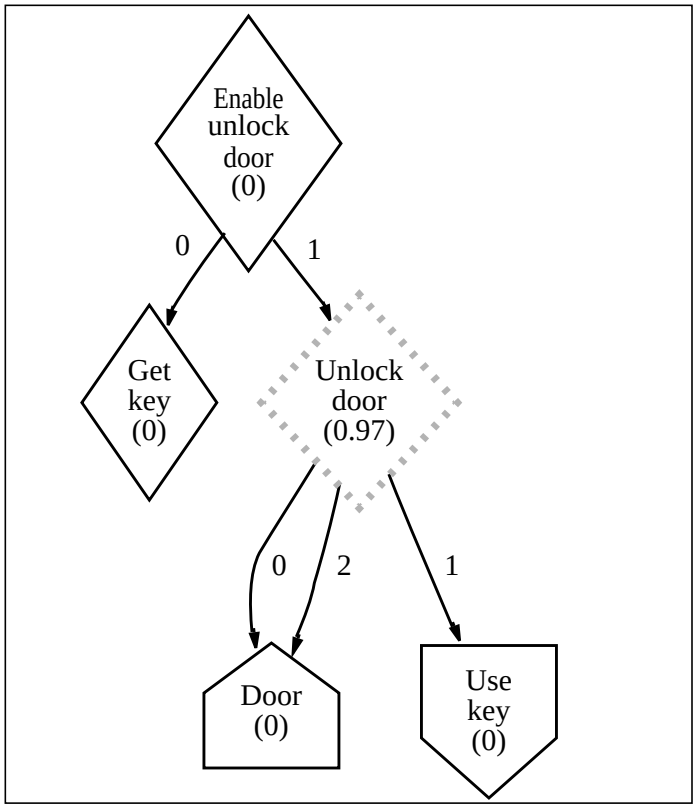

Figure 4 - Enable unlock door/Unlock door

The final two pieces are supplied in Figure 5: how to get a key (“Get key”), and how to get to the door from the world (“Go to door”).

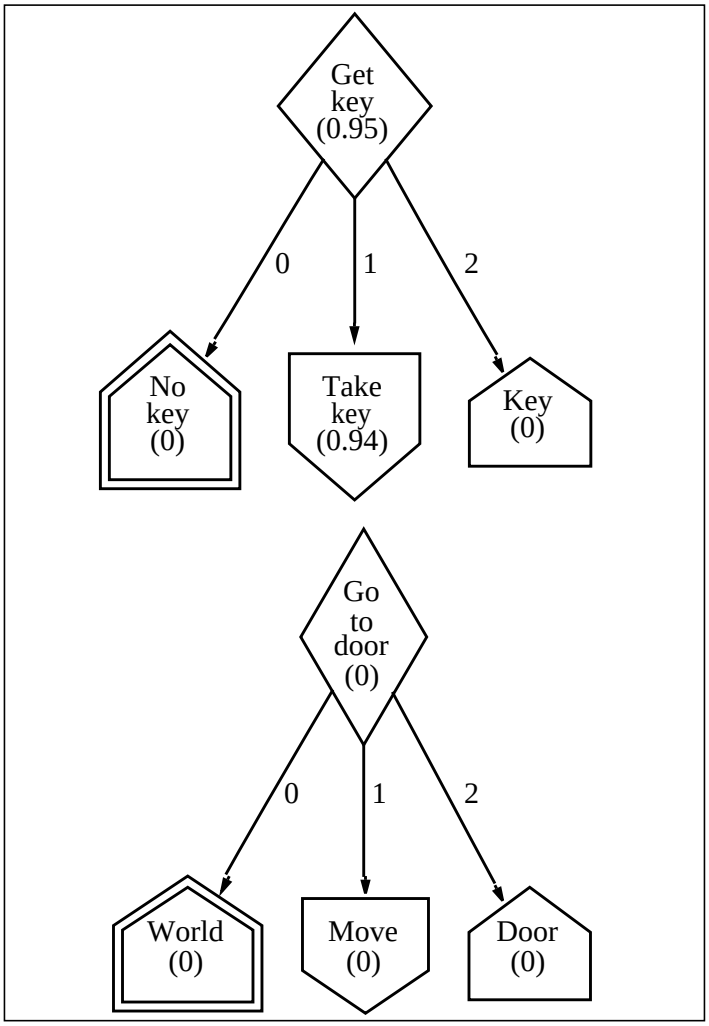

Figure 5 - Get key/Go to door

Since these diagrams show the initial state of network, the “World” and “No key” receptors are firing, denoted by the double outlines on their graphical symbols. Figure 6 shows the entire network, with each neuron type segregated into its own “cortex”.

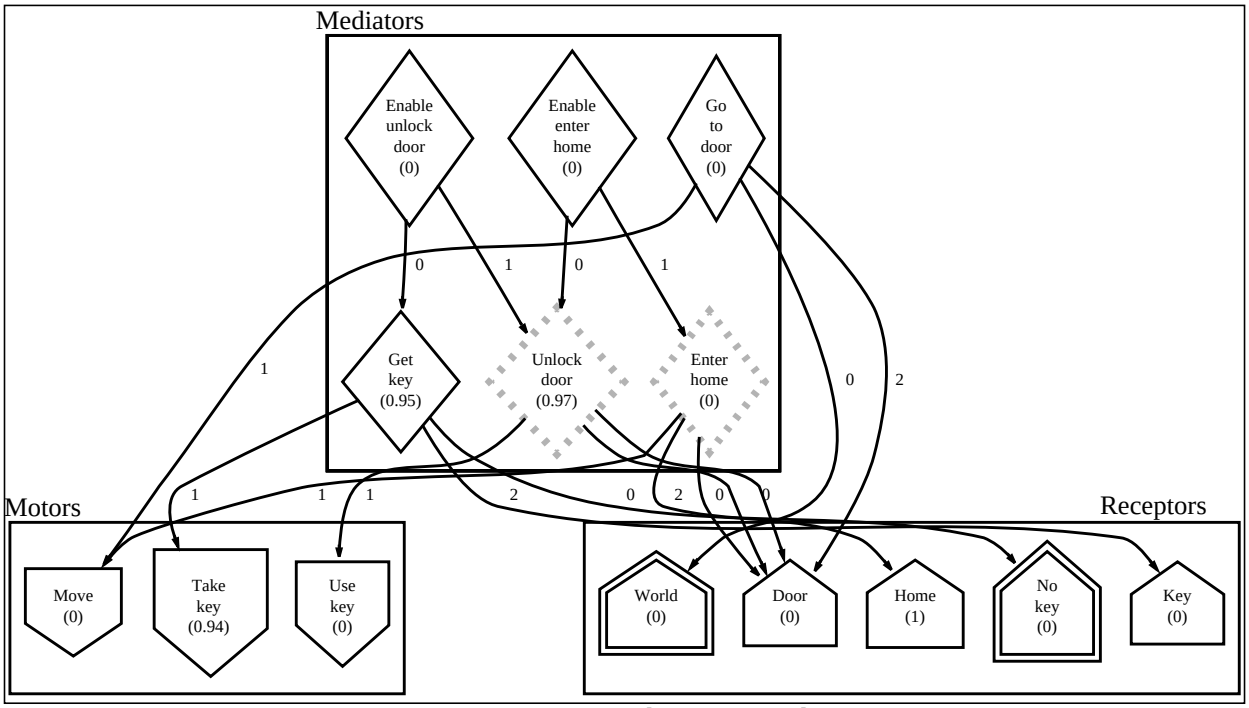

Figure 6- Getting home network

Neurons use a simple firing threshold function. Receptor and motor neurons fire when their associated sensory/response events occur. A mediator neuron contains an eventFiring() function, shown in Figure 7, which fires the mediator when each event in its sequence fires within the maximum delay imposed by the mediator's maxEventDelay value.

```
// Mediator event firing.
Mediator::eventFiring(eventNumber)
{
   // Discard unexpected events.
   if (eventNumber != expectedEvent) return;
   if (eventNumber == finalEvent)
   {
      // Final event firing: fire mediator.
      firing = TRUE;
      // Fire event in this neuron's mediators.
      for (all superMediator)
      {
         mediator = superMediator->mediator;
         event = superMediator->eventNumber;
         mediator->eventFiring(event);
      }
      // Reset event counter.
      expectedEvent = 0;
   } else {  // More events expected.
      // Expect next event.
      expectedEvent++;
      eventTimer = maxEventDelay;
      // If enabled, propagate enablement to expected event.
      if (enabled == TRUE)
      {
         neuron = components[expectedEvent];
         neuron->enabler += eventEnabler;
      }
   }
}
```

Figure 7 - eventFiring() function

When an event fires, if the mediator is enabled, a mediator-specific `eventEnabler` value is accumulated in the next event in the sequence, as shown in Figure 8.

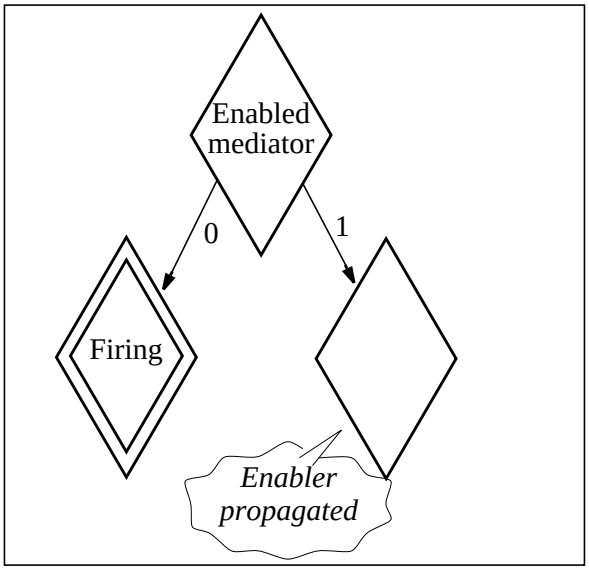

Figure 8 - Enabler propagation

Enabler propagation occurs when neurons fire; enablement represents a more persistent quantity. In the example, getting the key, a transient event, enables the ability to unlock the door, a persistent state. In addition, a neuron may be enabled by multiple mediators, each exerting enabling influences. These semantically different variables can be related in the following way. Let an enabler value be represented as a real number, positive denoting an enabling influence, negative a disabling one. Let enablement be boolean valued. Enablement can then be expressed by a hysteresis function of the sum of its enablers, as shown in Figure 9. A possible drawback to this function is that it may be less amenable to optimization techniques, e.g., an error-reduction algorithm.

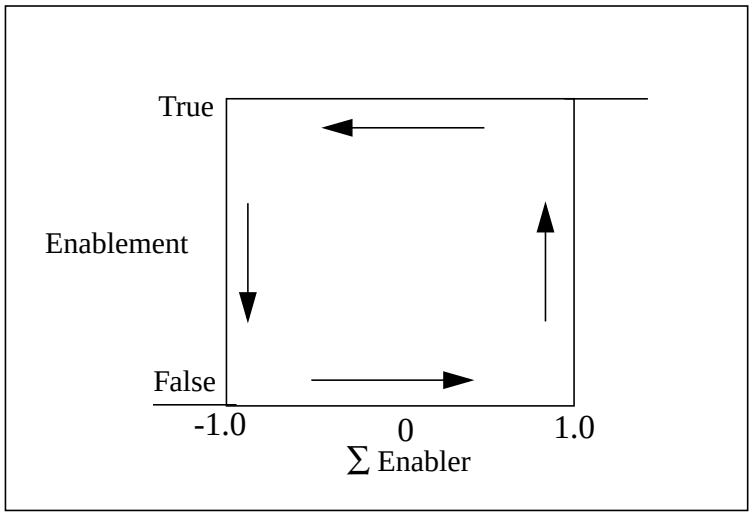

Figure 9 - Enabler/enablement hysteresis

Mona's *raison d'être* is need-reduction. For this purpose, some receptors are associated with the reduction of needs and are thereby defined to be goals. For example, a warmth receptor would be associated with a reduction of feeling cold. The drive() function, shown in Figure 10, propagates need from goal sources to other neurons in the network, attenuating to preferably drive "closer" neurons and to prevent endless propagation.

```
// Neuron drive.
Neuron::drive(need) {
{
   // Save maximum propagated need.
   if (need <= currentNeed) return;
   currentNeed = need;
   // Attenuate need.
   if ((need -= ATTENUATION) <= 0) return;
   // If this neuron is an enabled mediator,
   // drive its expected event.
   if (type == MEDIATOR && enabled == TRUE)
   {
      neuron = components[expectedEvent];
      neuron->drive(need);
      return;
   } else {
      // Drive this neuron's mediators.
      for (all superMediator)
      {
         mediator = superMediator->mediator;
         event = superMediator->eventNumber;
         mediator->eventDrive(event, need);
      }
   }
}
```

Figure 10 - drive() function

Need causes a mediator neuron to perform a check: if it is enabled, the eventDrive() function, shown in Figure 11, will pass the need into its expected event neuron in order to motivate it to occur, as shown in Figure 12.

```
// Mediator event drive.
Mediator::eventDrive(eventNumber, need)
{
   // Attenuate need.
   if ((need -= ATTENUATION) <= 0) return;
   // If this mediator is disabled, drive its mediators.
   if (enabled == FALSE)
   {
       for (all superMediator)
      {
         mediator = superMediator->mediator;
         event = superMediator->eventNumber;
         mediator->eventDrive(event, need);
      }
   } else {
      // Drive expected event.
      neuron = components[expectedEvent];
      neuron->drive(need);
   }
}
```

Figure 11 - eventDrive() function

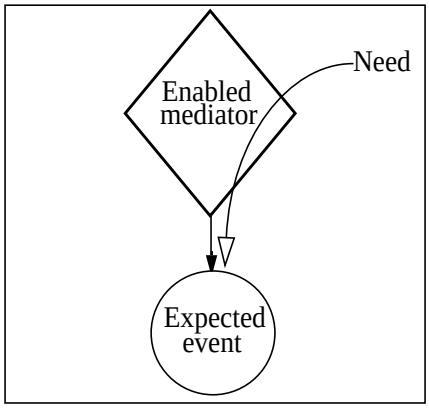

Figure 12 - Driving expected event

Otherwise, a disabled mediator (or a receptor or motor) drives its mediating neurons to cause them to enable it, as shown in Figure 13.

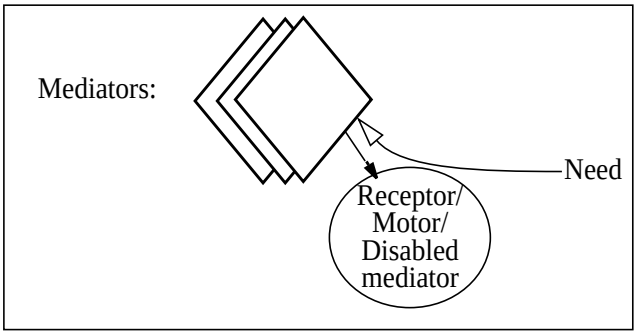

Figure 13 - Driving mediators

Upon completion of drive, the need resident in motor neurons is translated into potentials of the responses associated with those neurons. The system response is that associated with the maximum potential value:

$$\text{response}\left(\text{motor}_{\max(\Sigma\text{propagated need})}\right)$$

The complete algorithms, written in C++, are available on the World Wide Web (see Conclusion).

Table 1 contains a trace of firing neurons obtained when the “Getting Home” example is run on a software implementation:

Table 1: Firing neuron trace for “Getting home” network

time=0 Receptor firing: No key Receptor firing: World Motor firing: Take key ***time=1*** Receptor firing: Key Mediator firing: Get key Receptor firing: World Motor firing: Move ***time=2*** Receptor firing: Key Receptor firing: Door Mediator firing: Go to door Motor firing: Use key	***time=3*** Receptor firing: Key Receptor firing: Door Mediator firing: Unlock door Mediator firing: Enable unlock door Motor firing: Move ***time=4*** Receptor firing: Key Receptor firing: Home Mediator firing: Enter home Mediator firing: Enable enter home

2.3 The Network vs. a State-space Model

The network must embody a model of the environment which, for comparison purposes, can be sized against the well-known state-space model. In the network, the enabling and disabling operations allow the "topology" of the network to be modified by the act of operating it. The network may be considered to be a hybridization of a logic engine and a state-space search engine having two key properties: (1) more than one state (neuron) can be current (firing) at a particular time, and (2) current states can "prove" (enable) or "disprove" (disable) the reachability of states. These properties allow a network to assume a large number of states relative to the number of neurons which comprise it.

As an illustration, consider the personal financial state-space shown in Figure 14, in which money is earned at a job (pocket), spent at a store (broke), and deposited/withdrawn at a bank (saved). For the sake of simplicity, let there be a single quantum of money in the economy, e.g., if the money is in the bank, more money cannot be earned. The space contains (3 places) X (3 money situations) = (9 states).

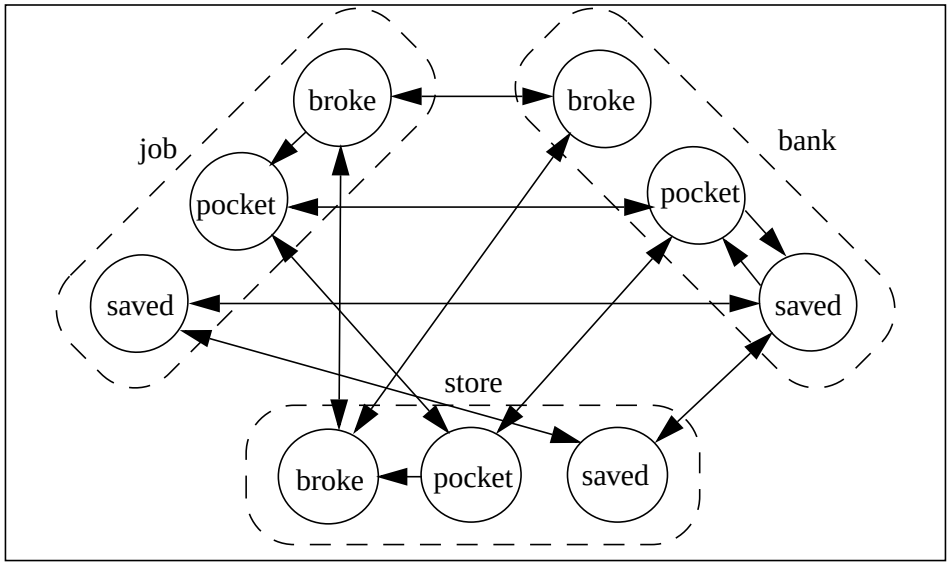

Figure 14 - Financial state-space

Figure 15 shows a network representation of the problem in abbreviated graphical form, for clarity. The ‘+’/’-’ indicate enabling/disabling influences. It can be seen that moving between places (the top portion of the figure) is independent of the transactions which transpire at those places since the enablement states (for earn, spend, deposit, and withdraw) store the transaction possibilities. The addition of places not involved in monetary dealings, a reasonable real-world supposition, alters only the place transition portion of the network, avoiding the combinatorial expansion of the state-space model. For example, if “home”, “park”, and “museum” are added as places, the state-space expands by 9 states, while the network expands by 3 neurons.

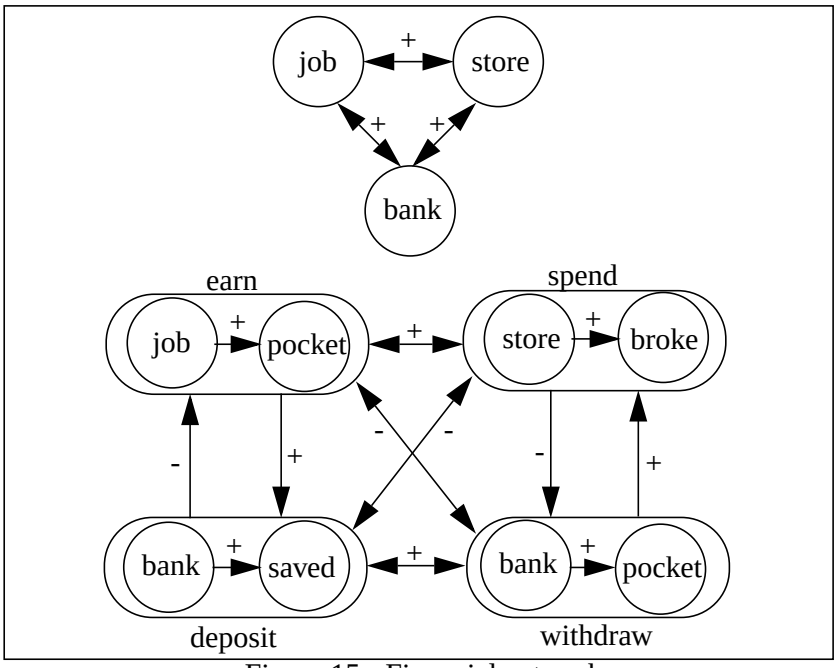

Figure 15 - Financial network

3 Demonstration

3.1 The Foraging Ant

This problem demonstrates how the neural network can be used to simulate a foraging ant. Foraging is a social enterprise among ants, which are known to follow trails of chemical markers left by each other to efficiently gather food (Bonabeau and Théraulaz, 2000). In this problem, the behavior of an individual ant performing one portion of the foraging process is simulated. The artificial ant must follow a meandering trail of marks from its nest to a piece of food, which it must then carry back to the nest. The problem illustrates the interplay of mediator neurons, using mutual enablement and disablement, to guide the ant to a goal in a semi-predictable environment.

A sample trail is shown in Figure 16. The ant starts at its nest and follows the trail marks to the cake. The trail is randomly generated in such a way that it never crosses itself. Generally a trail leads straight on, so the most efficient strategy is to plunge ahead and orient upon leaving the trail. An initial positive need is associated with the receptor which detects the presence of food at the nest.

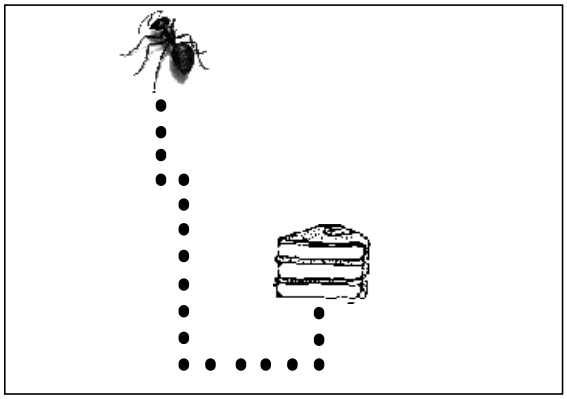

Figure 16 - Meandering ant trail

Sensory Capabilities.

- Presence of mark at current location.
- Presence of food at current location.
- Presence of nest at current location.

Response Capabilities.

Move forward and backward, Grab and drop food, Orient itself in the direction of the trail.

Need.

Food to be present at nest.

Mediators:

Format:
<mediator>:<“enabled”|”disabled”>/<“enabling”|”disabling”>(<event sequence>)

Top-level control: “Forage” is to “Get food” then enable “Bring food” to bring it back to the nest:
“Forage”: enabled/enabling (“Get food”,”Bring food”)
“Get food”: enabled/enabling (“Grab food”,”Orient”,”Mark”)
“Grab food”: enabled/enabling (“Food”,”Grab”,”No food”)
“Bring food”: disabled/enabling (“Nest”,”Drop”,”Food at nest”)

After food obtained, reset “Bring food” to disabled state for next forage:
“Disable bring food”: enabled/disabling (“Food at nest”,”Bring food”)

“Travel trail” is the normal way to move. “To food” and “To nest” associate “Travel trail” with getting somewhere:
“Travel trail”: enabled/enabling (“Mark”,”Forward”,”Mark”)
“To food”: enabled/enabling (“Travel trail”,”Food”)
“To nest”: enabled/enabling (“Travel trail”,”Nest”)

If the ant steps off the trail (“No mark”), these control getting it back on:
1. Disable normal traveling (“Travel trail”).
2. Initiate “Trail search” to back-up and orient.
3. Once oriented, enable normal traveling.
“Disable travel trail”: enabled/disabling (“No mark”,”Travel trail”)
“Trail search”: enabled/enabling (“No mark”,”Backward”,”Mark”,”Orient”,”Mark”)
“Enable travel trail”: enabled/enabling (“Trail search”,”Travel trail”)

Figure 17 shows the entire network:

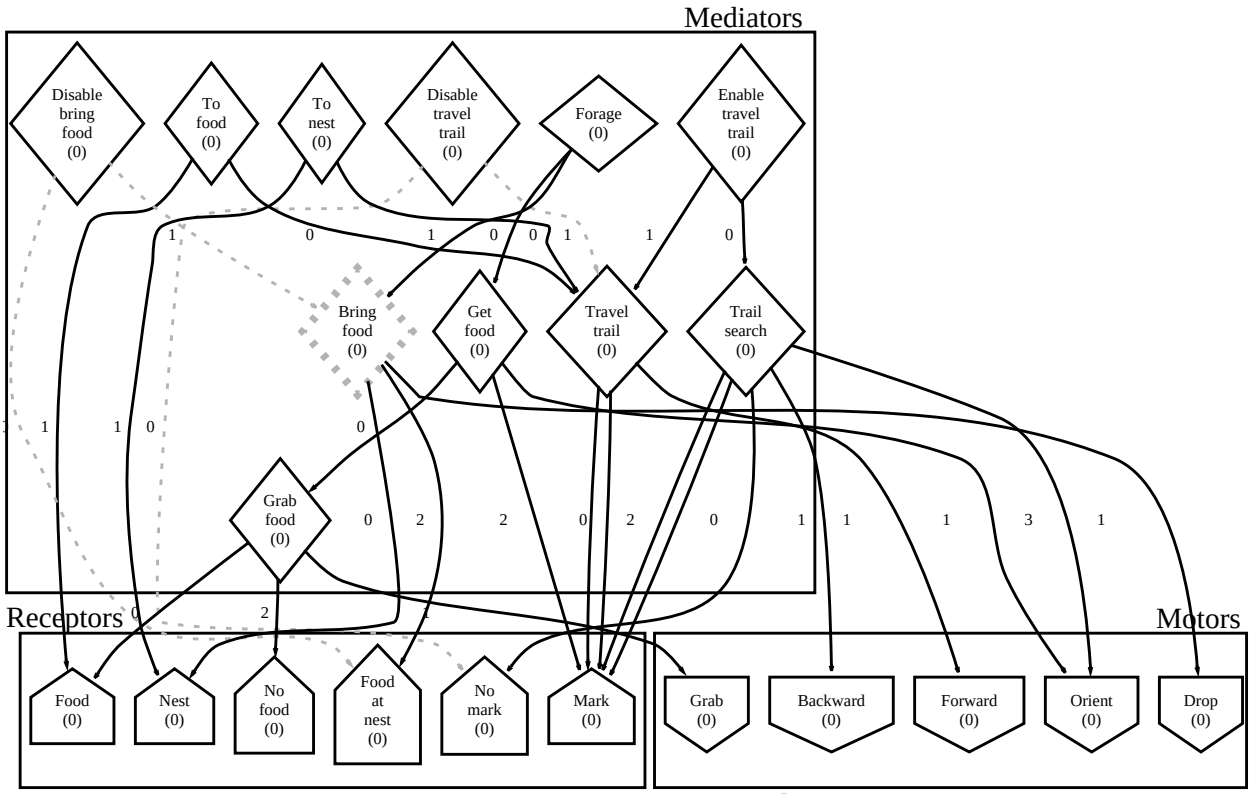

Figure 17 - Foraging ant network

The initial response of the ant is to move forward, driven by the need to fetch food to the nest, as shown in Figure 18:

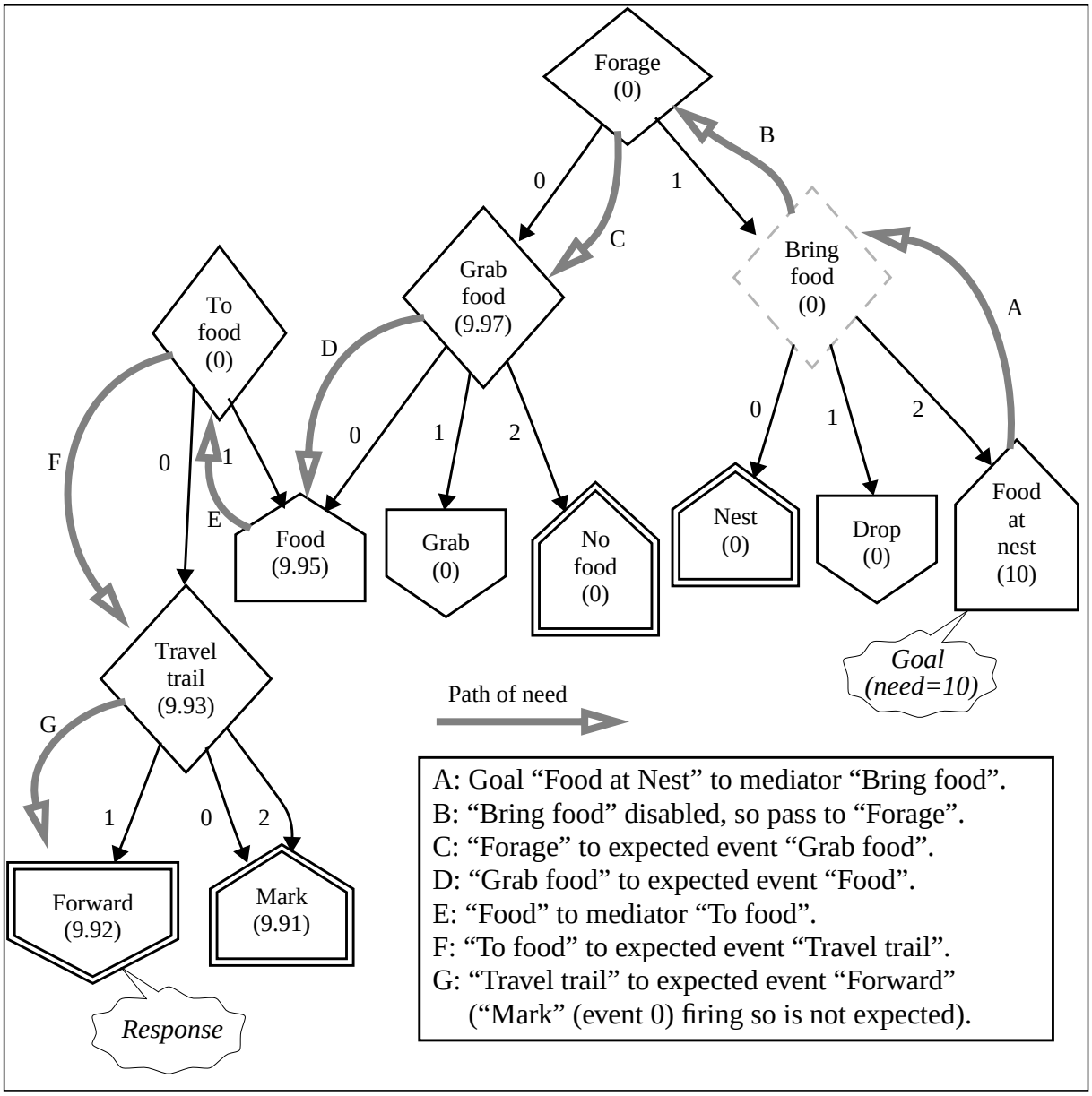

Figure 18 - Driving initial foraging response

Running the program on the sample trail results in the responses listed in Table 2 to fetch the food to the nest:

Table 2: Foraging responses

Forward X 5 Backward Orient Forward X 2 Backward Orient Forward X 8 Backward Orient Forward X 6 Backward Orient Forward X 3 Grab	Orient Forward X 4 Backward Orient Forward X 6 Backward Orient Forward X 8 Backward Orient Forward X 2 Backward Orient Forward X 4 Drop

Of particular interest is the interplay of neurons involved in orienting the ant after stepping off the trail, shown in Table 3:

Table 3: Firing neuron trace for trail orientation

time=n Receptor firing: Mark Receptor firing: No food Mediator firing: Travel trail Motor firing: Forward	Ant on trail.
time=n+1 Receptor firing: No mark Receptor firing: No food Motor firing: Backward	Ant steps off trail: • "Disable travel trail" disables "Travel trail". • "Trail search" mediates moving backward.
time=n+2 Receptor firing: Mark Receptor firing: No food Motor firing: Orient	Ant now orients to trail: "Trail search" mediates orient response.
time=n+3 Receptor firing: Mark Receptor firing: No food Mediator firing: Trail search Motor firing: Forward	"Trail search fires", allowing "Enable travel trail" to re-enable "Travel trail". Ant moves forward in correct direction.

3.2 The Nesting Birds

In this task, a pair of nesting birds are simulated. The birds are the ground-nesting sort, and live in a cellular world having three terrain regions: grassland, forest, and desert. Beginning together in the grassland, the birds build their rectangular nest using stones found in the desert, and once constructed, the female lays an egg in it. They periodically require food in the form of mice which are found in the forest.

Gender-specific Roles.

Male roles:

- Find food and feed self when hungry.
- Find and fetch stones for female when requested by her.
- Find and fetch food for female when she is hungry.
- Stay by female when not busy.

Female roles:

- Signal need for food to mate when hungry and feed when he provides food.
- Build nest: (1) request and accept stones from mate, and (2) place stones in rectangular configuration.
- Repair nest: (1) replacing missing stones, and (2) removing extraneous stones.
- Lay egg in completed nest.

Sensory Capabilities.

- Terrain at current location.
- Object at current location.
- Condition: wanting food, wanting stone, and object being held.
- Condition of mate if co-located.

Response Capabilities.

Do nothing, Eat, Get (object), Go to desert, Go to forest, Go to mate, Lay egg, Look for food, Look for stone, Put (object), Receive (object), Step, Toss (object), Turn, Want stone, Do not want stone.

The “Go to” responses cause the bird to step toward the nearest cell with the indicated terrain or mate. An innate ability to determine the correct direction is assumed. The “Look for” responses likewise step the bird to the indicated object, but only if the bird is in the correct terrain. The “Want stone” and “Do not want stone” responses change the condition of the bird. The “Toss” response causes any held object to be discarded to a random nearby location.

Gender-specific Needs.

Needs are listed in order of high to low precedence.

Male needs:

1. Food:
   Initiated periodically.
   Satisfied by eating.
2. Female needs food:
   Initiated by presence of female wanting food.
   Satisfied by presence of female not wanting food.
3. Female needs stone:

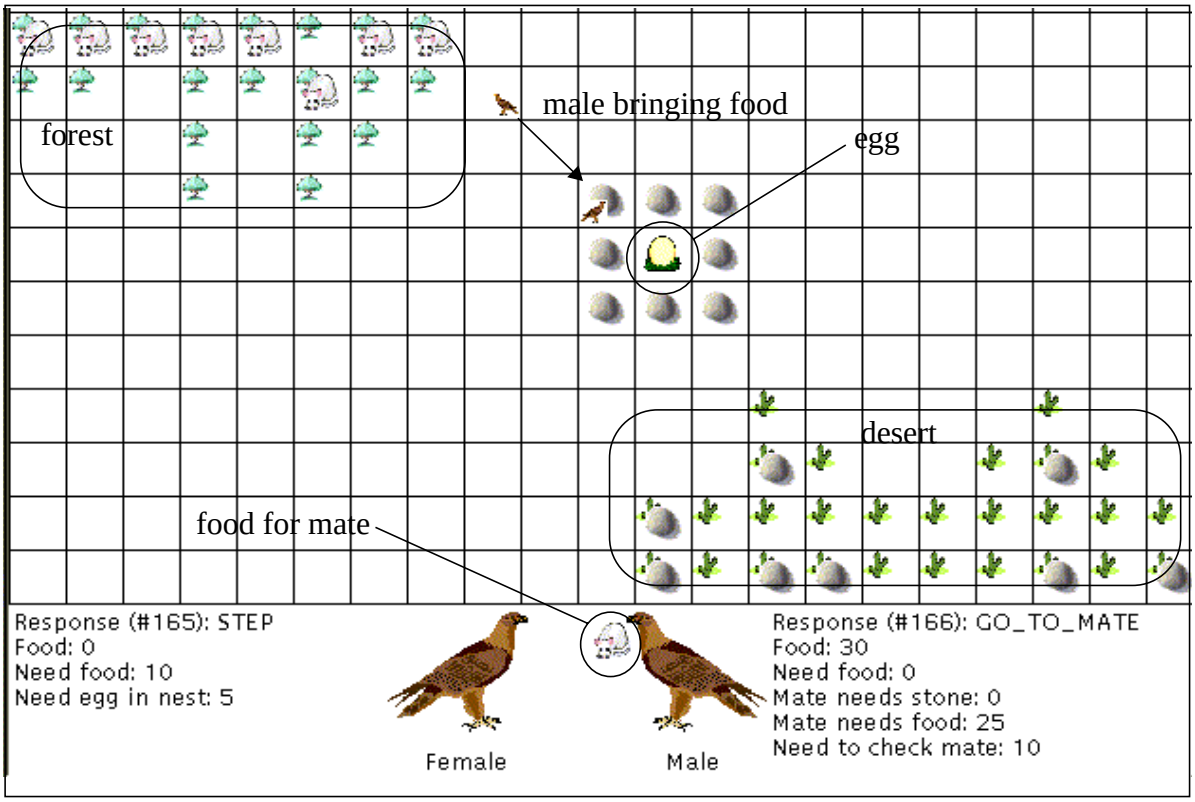

Figure 19 - Nesting birds snapshot

Initiated by presence of female wanting stone.
Satisfied by presence of female not wanting stone.

4. Stay by female: a constant need.

Female needs:

1. Food:
   Initiated periodically.
   Satisfied by eating.
2. Egg in nest:
   A constant need, prompting not only building of initial nest and laying of egg in it, but also vigilance in the repairing nest and replacing a missing egg.

Figure 19 shows an abbreviated snapshot of the end of an animation of the nesting birds program. The male is bringing a mouse back to the hungry female, who minds the nest and egg. The complete animation can be seen in its entirety at http://www.corecomm.net/portegys/NestViewer.html (JAVA applet).

Gender-specific Mediators.

Format:
<mediator>:<“enabled”|”disabled”>/<“enabling”|”disabling”>(<event sequence>)

Male mediators:

Food:

“Eat food”:enabled/enabling(”Got food”,”Eat”,”Not hungry”)
“Get food”:enabled/enabling(”Food on ground”,”Get”,”Got food”)
“Find food”:disabled/enabling(”Any terrain”,”Look for food”,”Food on ground”)
“Find forest”:enabled/enabling(”Any terrain”,”Go to forest”,”Forest”)
“Enable find food”:enabled/enabling(”Find forest”,”Find food”) # Once in forest, can find food
“Disable find food”:enabled/disabling(”Find food”,”Find food”) # To force re-finding of forest first

Stones:

“Get stone”disabled/enabling(”Stone on ground”,”Get”,”Got stone”)
“Find stone”:enabled/enabling(”Any terrain”,”Look for stone”,”Stone on ground”)
“Find desert”:enabled/enabling(”Any terrain”,”Go to desert”,”Desert”)
“Enable get stone”:enabled/enabling(”Find desert”,”Find stone”) # Once in desert, can find stone
“Disable find stone”:enabled/disabling(”Find stone”,”Find stone”) # To force re-finding of desert first

Coordinators:

“Get food disables get stone”:enabled/disabling(”Get food”,”Get stone”) # Can hold only one object
“Get stone disables get food”:enabled/disabling(”Get stone”,”Get food”) # Can hold only one object
“Toss object”:enabled/enabling(”Any terrain”,”Toss”,”Got no object”) # Have no object after toss
“Toss enables get food”:enabled/enabling(”Toss object”,”Get food”) # Can hold food after toss
“Toss enables get stone”:enabled/enabling(”Toss object”,”Get stone”) # Can hold stone after toss
“Toss disables find stone”:enabled/disabling(”Toss object”,”Find stone”) # Must find desert first
“Toss disables find food”:enabled/disabling(”Toss object”,”Find food”) # Must find forest first

Mate:

“Feed female”:enabled/enabling(”Got food”,”Go to mate”,”Mate has food”,”Do nothing”,”Mate does not want food”)

“Give stone to female”:enabled/enabling(”Got stone”,”Go to mate”,”Mate has stone”,”Do nothing”,”Mate does not want stone”)
“Check on mate”:enabled/enabling(”Any terrain”,”Go to mate”,”Mate present”)

Female mediators:

Food:
“Eat food”:enabled/enabling(”Got food”,”Eat”,”Not hungry”)
“Receive food”:enabled/enabling(”Mate has food”,”Receive”,”Got food”)

Nest building:
“Egg in nest”:enabled/enabling(”Stone on ground”,”Step”,
”Stone on ground”,”Turn”,”Stone on ground”,”Step”,”Stone on ground”,”Step”,
”Stone on ground”,”Turn”,”Stone on ground”,”Step”,”Stone on ground”,”Step”,
”Stone on ground”,”Turn”,”Stone on ground”,”Step”,”Stone on ground”,”Step”,
”Stone on ground”,”Turn”,”Stone on ground”,”Step”,”Stone on ground”,”Turn”,
”Stone on ground”,”Step”,”Egg on ground”) # Sequence for constructed nest

“Build nest”:enabled/enabling(”Empty ground”,”Want stone”,
”Mate has stone”,”Receive”,”Have stone”,”Do not want stone”,”Have stone”,”Put”,
”Stone on ground”) # To request and place a missing stone

“Clear nest”:enabled/enabling(”Stone on ground”,”Get”,”Have stone”,”Toss”,”Ready to lay egg”) # To remove extraneous stone

“Begin nest check”:enabled/enabling(”Any terrain”,”Step”,”Any terrain”,”Turn”,”Any terrain”) # Must step out of nest to restart egg in nest sequence

Egg laying:
“Lay egg in nest”:enabled/enabling(”Ready to lay egg”,”Lay egg”,”Egg on ground”)
“Disable egg in nest”:enabled/disabling(”Egg in nest”,”Egg in nest”) # Forces bird out for nest checking
“Enable egg in nest”:enabled/enabling(”Begin nest check”,”Egg in nest”) # Can re-lay egg if necessary after checking nest

Graphical views of the entire networks are omitted for space reasons.

Analysis.

Several scenarios in the nesting process are cited to illustrate the workings of the neural networks.

1. Female requests food, male goes to forest, fetches food and gives to female to eat.
   Context: Female needs food, raising want food condition. Male, co-located with female, senses want food condition of female which raises need to feed her.
   Responses: See Table 4.

Table 4: Food fetching responses

Male	Female
Go to forest	Want stone
For male, “Feed female” drives “Get food” which drives “Find food” which drives “Find forest”. Female also wants stone for nest, but food takes precedence.	
Go to forest	Want stone
Look for food	Want stone
“Find forest” fires, enabling “Find food”.	
Get	Want stone
Go to mate	Want stone
Go to mate	Receive
Do nothing	Eat
“Mate does not want food” fires, lowering need to feed female.	

2. While building nest, female requests stone, male goes to desert, fetches stone and gives to female, who places it in nest.
   Context: The “Egg in nest” mediator drives the female to step around the perimeter of the nest, observing stones as she goes. “Egg in nest” itself does not mediate the placement of stones, but employs another mediator, “Build nest”, to do this by transferring need to it.
   Responses: See Table 5.

Table 5: Stone fetching responses

Male	Female
Go to mate	Want stone
Male has obtained stone and is returning.	
Go to mate	Receive
Do nothing	Do not want stone
Do nothing	Put
“Build nest” fires, “Egg in nest” resumes.	
Do nothing	Step
Go to mate	Want stone
“Build nest” repeated; male instilled with need to fetch another stone.	
Go to desert	Want stone

3. Male becomes hungry while returning with stone, tosses stone and goes for food.
   Context: The need to food overrides the need to fetch a stone. The “Toss enables get food” mediator enables “Get food” by freeing the bird to pick up food. After eating, “Give stone to female” reasserts itself, and the male proceeds to fetch another stone from the desert.

Responses: See animation.

4 Related Work

In contrast with Hopfield and backpropagating ANNs (Hopfield and Tank, 1986; Munakata 1998), which are primarily stateless pattern classifiers, Mona employs a short term memory capability to navigate the environment toward goals. Short term memory is implemented by the retention of neural firing sequences and by the enabling and disabling operations, which modify the state of the network based on sensory events and responses motivated by needs. The incorporation of state memory into ANNs is a topic of continuing investigation (Roy 1997; Kodjabachian and Meyer, 1998)

A large body of cross-disciplinary work on the subject of animal and animat social behavior exists (Goss and Deneubourg, 1992; Drogoul and Ferber, 1993; Anderson, Blackwell, and Cannings, 1997; Bonabeau, Dorigo, and Théraulaz, 1999; etc.). Reynolds' (1987) much-publicized simulation of bird flocking, Boids, showed how several simple rules applied by individual birds combined to produce emergent group behavior. In the multiagent system developed by Murciano and Millán (1996), agents cooperate and specialize to perform a collective gathering task by communicating relevant location data using light signals. Behavioral parameters were subject to post-trial learning to improve performance. Mataric (1995) produced various emergent group behaviors, such as herding, by combining more elementary forms. For example, "herding" behavior consists of a combination of "flocking" and "surrounding".

For the three systems cited above, while interesting group behavior is exhibited on the assigned tasks, the individual agents are programmatically defined with task-dependent knowledge, and thus limited to use in narrow domains. One of the goals in this work is to investigate the behavior of agents constructed from task-independent components. Different configurations of such components result in agents capable of performing varied tasks. In Mona, networks are constructed of three types of neurons: receptors, motors, and mediators. Within a network, neurons of a given type are distinguishable only by their internal parameters and interconnections. A future goal is to design systems which can re-configure their components by learning or evolution, and thereby become capable of performing new tasks.

5 Conclusions

Real birds and ants must deal with more numerous and complex needs than the simulated creatures presented in this paper, of course. In addition to a more complex nest-building process, birds must also regulate egg temperature, rear hatchlings, fend off predators, etc. What is of primary interest here is the general question of how artificial neural networks can produce lifelike behavior. A basic assumption is that behavior is motivated by needs which are translated into responses. Using the model, the ant successfully forages for food in a class of simulated environments, and the pair of simulated birds are able to cooperate in the construction of a nest while keeping themselves fed, a process involving the orchestration of sometimes conflicting needs.

The programming for the Mona neural network model and several tasks, including the nesting birds and foraging ant, may be downloaded from http://www.corecomm.net/portegys/ai.html.

6 Future Work

6.1 Context Dependency

In Schank's scripts (Schank and Childers, 1984), environmental cues combine to produce expectations about available scenarios. For example, the presence of a table, crowd noise, and waiter

contribute to the expectation of things available in a restaurant, such as the ability to order and eat a meal. Cues form intersections of contexts, yielding expectations. Scripts are a powerful model of cognition at a high level of abstraction: that of natural language. In the terminology of Mona, the abstraction becomes grounded in a neural network; the firing of events associated with the table, crowd noise, and waiter accumulate to enable mediators of restaurant events. A unique context of events can also serve to discriminate an instance of a general type. For example, I want to relate to my dog both as my pet (discrimination) and as a canine (generalization). Categorizing the environment in a contextual way is a vital part of cognition and clearly necessary for effective goal-seeking.

6.2 Learning

Although Mona adapts to its environment using short term memory, it obviously lacks a learning or long-term adaptation capability, which necessitates manual composition of the network. This omission exists in order to initially focus on operational aspects of the model. The general applicability of the model will only be realized when learning is added. This would likely involve a “hypothesizer” function, which observes sequences of events and posits causality between them. Mona is amenable to this type of learning: receptor neurons embody sensory events; motor neurons embody response events, and mediator neurons embody event sequences. Successful hypotheses would strengthen mediators, while failing ones would weaken them.

7 References

Anderson, C., Blackwell, P. G., and Cannings, C. (1997). Stochastic Simulation of Ants that Forage by Expectation. In P. Husbands and I. Harvey (Eds.), *Fourth European Conference on Artificial Life*. Cambridge, MA: MIT Press.

Albus, J. S. (1979). Mechanisms of Planning and Problem Solving in the Brain. *Math. Biosci.* 45, 247-293.

Bonabeau, E., Dorigo, M., and Théraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.

Bonabeau, E. and Théraulaz, G. (2000). Swarm Smarts. *Scientific American*. 282:3, 72-79.

Drogoul, A. and Ferber, J. (1993). From Tom Thumb to the Dockers: Some Experiments with Foraging Robots. In J-A Meyer, H. L. Roitblat, and S. W. Wilson (Eds.), *From Animals to Animats II: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.* Cambridge, MA: MIT Press.

Dudai, Y. (1989). *The Neurobiology of Memory.* New York: Oxford University Press.

Fu, L. (1994). *Neural Networks in Computer Intelligence*. McGraw-Hill, Inc.

Goss, S. and Deneubourg, J.L. (1992). Harvesting by a Group of Robots. In F.J. Varela and P. Bourgine (Eds.), *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life.* Cambridge, MA: MIT Press.

Hampson, S. E. (1990). *Connectionist Problem Solving: Computational Aspects of Biological Learning*. Birkhäuser Boston.

Holmes, W. and Rall, W. (1992) Electrotonic Models of Neuronal Dendrites and Single Neuron Computation. In T. McKenna et al. (Eds.), *Single Neuron Computation*, San Diego, CA: Academic Press.

Hopfield, J. and Tank, D. (1986). Computing with Neural Circuits: A Model. *Science*, 233, 625-633.

Kodjabachian, J. and Meyer, J-A. (1998). Evolution and Development of Neural Controllers for Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects. *IEEE Transactions on Neural Networks.* 9:5, 796-812.

Mataric, M. (1995). Designing and Understanding Adaptive Group Behavior. *Adaptive Behavior.* 4:1, 51-80.

McClelland, D. (1987). *Human Motivation*. Cambridge: Cambridge University Press.

Munakata, T. (1998). *Fundamentals of the New Artificial Intelligence: Beyond Traditional Paradigms*. New York: Springer-Verlag Inc.

Murciano, A. and Millán, J. (1996). Learning Signaling Behaviors and Specialization in Cooperative Agents. *Adaptive Behavior.* 5:1, 5-28.

Nolfi, S. and Parisi, D. (1996). Learning to Adapt to Changing Environments in Evolving Neural Networks. *Adaptive Behavior.* 5:1, 75-98.

Parten, C. R. (1990). *Handbook of Neural Computing Applications*. Academic Press, Inc.

Pfaff, D. W. (1982). Motivational Concepts: Definitions and Distinctions. In E. Pfaff (Ed.), *The Physiological Mechanisms of Motivation.* New York: Springer-Verlag Inc.

Portegys, T. (1986) *GIL - an Experiment in Goal-directed Inductive Learning*. Ph.D. dissertation, Northwestern University, Evanston (Available from UMI at http://www.umi.com/).

Portegys, T. (1999). A Connectionist Model of Motivation. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99).* IEEE Catalog Number: 99CH36339C.

Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavior Model. *Computer*

Graphics 21:4, 25-34.
Roy, A. (1997) Panel Discussion at ICNN97 on Connectionist Learning. In D. Levine (Ed.), *Neural Networks*, 2:2.
Schank, R. C., with Childers, P. G. (1984). *The Cognitive Computer; On Language, Learning, and Artificial Intelligence*. Addison-Wesley Publishing Company, Inc.
Skinner, B. F. (1957). *Verbal Behavior.* New York: Appleton-Century-Crofts.
Thompson, R., Berger, T., and Berry S. (1980). Brain Anatomy and Function. In M. Wittrock (Ed.), *The Brain and Psychology,* Academic Press.
Waltz, D. (1999). The Importance of Importance. *AI Magazine*. 20:3, 18-35.