Learning Motivational Structures Using Neural Schemas

Lee McCauley
The University of Memphis
t-mccauley@memphis.edu [1]

Humans live in a sea of competing and complementary motivations. One reason that our behaviour is so complex and interesting stems from the hundreds of separate desires and goals that influence our actions. But we do not come into this world with these complex and often subtle motivators. Instead, humans learn to achieve or maintain intermediate goals that tend to facilitate the achievement of our base-level drives. We amass wealth, for example, because it facilitates the satiation of other drives such as hunger, security, or even procreation. Even though it can often be the case that a person will focus on the intermediate goal rather than the true priorities, these sub-goals are usually beneficial. This paper describes a new autonomous agent control structure, called neural schema mechanism, that learns through its own experience which intermediate states or goals should be achieved or avoided based on its primitive drives. The mechanism can also learn sensory motor correlations and abstract concepts. In addition, a psychological theory of consciousness is modelled that allows the system to come up with creative action sequences to achieve goals even under situations of incomplete knowledge. The result is an architecture for robust action selection that learns not only how to achieve primitive drives, but also learns appropriate sub-goals that are in service of those drives.

Keywords: Schema, learning, motivation, consciousness, neural

[1] *Address for correspondence: The University of Memphis, Dept. of Mathematical Sciences, Dunn Hall Rm. 373, Memphis, TN 38152*

Introduction

As soon as a child is born they are given the daunting task of creating a reality. They must form the knowledge and goal structures that will dictate how they interact with the world around them – and why. From relatively few initial clues each child must learn everything that they need to survive and prosper. They need to learn things like how their muscles work, how to communicate with their parents and others, and how to conform to the norms of their society. Some of this learning is just a matter of noting the common sensory results to actions performed under a given situation, simple muscle control, for example. Almost all learning beyond these first simple steps, however, requires that the child be able to judge not only what an action does, but also whether that result is a desirable one or not. Luckily for humanity, evolution has provided us with a set of innate sensory inputs that are pre-wired in our brains to give us pleasure, pain, happiness, sadness or any number of other feelings. Even so, there is more to what a child must learn than just which actions directly result in activation of one or more of the innate feelings; they must also learn which environmental states that, in and of themselves, do not trigger an innate response, nonetheless represent an increased likelihood of encountering a state that does trigger an innate positive or negative response at some point in the future. For humans, these intermediate states will eventually begin to trigger some of the feelings previously only activated upon attaining the future state. For example, a toddler, known well to the author, noting that her meals are always served while she is in her highchair, takes it upon herself to climb into her highchair when she is hungry. While this is a simple, almost Pavlovian, example it is, nonetheless, indicative of the fact that the intermediate state, which is specific to her environment, has become a sub-goal of the originally desired state. More complex and subtle examples might include the feeling you get when you see your pay stub. It is not very satisfying to eat that piece of paper and it wouldn't do much in the way of protecting you against the elements should you be trapped in the rain or cold, but you will still have a good feeling when you look at it.[2]

Without these feelings to pull us towards achieving (or avoiding) the intermediate states we might never achieve (or avoid) the original innate states. Taking inspiration from human psychology, the neural

[2] Ok, maybe some of us also get a sinking feeling, but the point still holds.

schema mechanism, described in this paper, makes advances toward a robust agent control structure that learns about its environment in several different ways including the creation of intermediate goal structures that, over time, will acquire appropriate motivational values based on the agent's experience.

To make the first steps in this direction a sufficiently general mechanism called schema mechanism created by Gary Drescher (1985, 1987, 1991) was extended to more closely approximate some human cognitive phenomenon. The new extension is called neural schema mechanism due to its resemblance to connectionist architectures. One could envision neural schema mechanism as a network of interconnected nodes looking very much like a neural network. Some of the nodes in this network represent states of the environment, called item nodes; others, action nodes, represent actions that can be taken by the agent. Still others, called schema nodes, represent knowledge about the relationships between the environmental states and the actions. Finally, goal nodes, are present that act as the motivators of the system, trying to influence how activation is spread through the network in order to bring about or avoid bringing about the environmental state that they represent.

All of these nodes are connected through various kinds of links. The many different types of links each have slightly different meanings and functions; some, such as action or goal-type links, simply serve as information pathways. Others, such as the generic type "none" links, serve as learning tools, collecting statistical information about the relationships between the nodes to which they are connected. In fact, it is a bit misleading to say that the schema nodes, alone, represent the knowledge of the system; it would be better stated to say that the schema nodes along with their result and context-type links actually describe the knowledge that is centered on the schema node. The upshot of all this is a mechanism that uses its own experience to learn about its world by creating new nodes and links that embody new knowledge about its environment and its own abilities to effect that environment. In addition, the mechanism includes an attention and "consciousness" module that allows it to more efficiently utilize resources and discover novel and original solutions to daunting problems. In later sections neural schema mechanism will be described in detail using simple sensory and motor modules as examples. Such modules, however, could well be very complex. Any of the advanced techniques from other more specific AI research could be used as senses or actuators.

All told, the neural schema mechanism takes steps toward the implementation of a comprehensive mechanism of mind that includes conceptual and sensory-motor learning, "consciousness," and a blending of logic with desires via a dynamic motivational system.

Neural schemas

The new implementation of schema mechanism is called a neural schema mechanism because of its translation of the original structures into node and link elements[3]. This section will detail the neural schema mechanism and how its structures interact to achieve robust learning and goal directed action.

In general, one can think of the mechanism as a network of nodes connected via links (see Figure

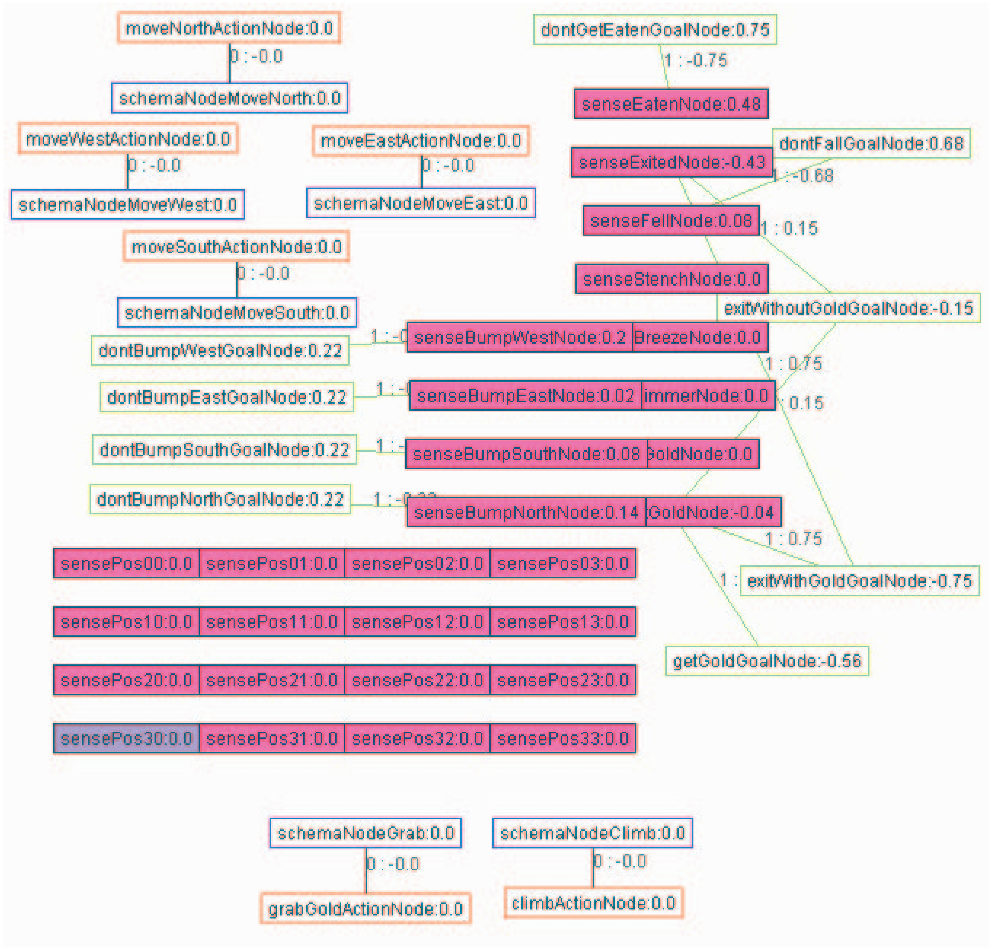

Figure 1: Initial state of a neural schema network for the Wumpus World environment.

[3] Drescher actually views the schema mechanism as a connectionist network but only to the degree that the schema slots can be interpreted as links; there is no spreading of activation.

5). Each node has its own activation, which it can spread to other nodes through the links. The links perform a function on the activation sent through them from their input node and deliver the resulting activation to their output node. With the exception of some rules that dictate which links are allowed to transmit activation, this part of the mechanism works exactly like any other connectionist network. Abilities are added to the nodes and links of the network that allow them to keep track of the necessary statistical data for the schema mechanism. In addition, nodes have the ability to check the statistical information in order to determine if a new node needs to be created and have the ability to create the new network elements. The system also contains the functionality to approximate a psychological theory of consciousness. In addition to providing a stronger psychological foundation, this allows the mechanism to discover novel solutions to otherwise insurmountable situations.

Throughout the course of this paper, examples and illustrations from a sample environment will be used where appropriate. The environment chosen for these examples and used for some of the experimentation of the mechanism is called Wumpus World (Figure 2) borrowed from Russell and Norvig (1995). Figure 1 shows an example of a neural schema network for the Wumpus World in its initial state. The names of the nodes at this point accurately reflect their purpose. To summarize, however, the agent, in this case the hunter, has the following percepts if he is within one square of a given object: the glitter of gold, the breeze from a pit, and the stench of the wumpus. The hunter can also sense the following percepts

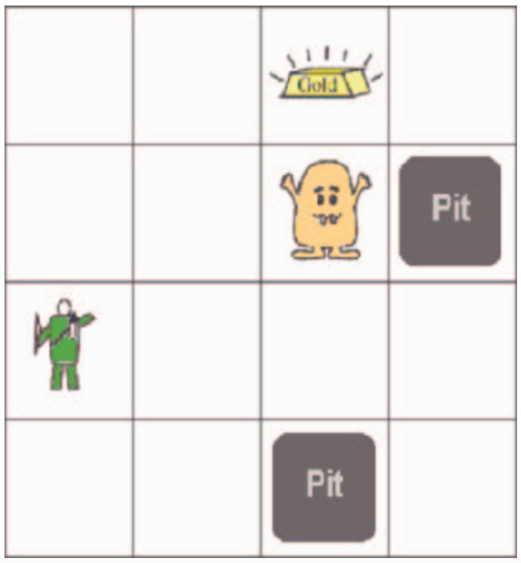

Figure 2: Wumpus World (Russell and Norvig 1995)

if he is within the same square as a given item: the gold, being eaten by the wumpus, and falling in a pit. The agent also senses his position, whether his is carrying the gold, if he bumps into a wall (directionally significant), and if he has exited the cave (grid). The hunter has the ability to move in any of the four directions (north, south, east or west), grab for the gold, and climb out of the cave. The objective for the agent is to locate the gold and find his way out without being eaten by the wumpus or falling into a pit. To reflect these overall objectives, goal nodes attach to the appropriate sensory item nodes and denote the relative like or dislike of having that item on or off. In addition to the goals directly related to the objectives, the hunter also has a mild dislike for running into walls (the edges of the grid).

Links

In the neural schema mechanism, links perform all the same functions as you would expect links to perform in a connectionist network. Even though each link is of a specific type, the workings of every link is, for the most part, the same; the type labels are used by nodes in the system to determine if it needs to deal with that link in some special way. The link types used in the current implementation are *context*, *result*, *goal*, *action*, *host*, and *none*. Table 1 lists the different types of links and gives a brief description of its key functions. Parts of the descriptions may not make a lot of sense at this point, but the table should come in handy as a quick reference for future sections.

Table 1: Link types

Link Type	Description	Primary Purpose
Context	Connects an input Item node with an output Schema node	Marks an Item node in a given state (*on* or *off*) as having a correlation with the successful execution of the Schema node
Result	Connects an input Schema node with an output Item node	Marks that the Item node transitions to a given state (*on* or *off*) when the Schema node is executed
Action	Connects an input Schema node with its one associated Action node or connects a learned input Action node to its learned output Goal node	Provides a path through which an Action node can send information back to its calling Schema node(s) or provides a path through which a learned Goal node can send information back to its calling learned Action node
Goal	Connects an input Item node with an output Goal node	Marks the state of the Item node that the Goal node is trying to achieve or avoid and provides a path through which resistance can be sent from the Goal node
Host	Connects an input learned Item node with an output Schema node	Mark those Schema nodes from which the learned Item node derives its state

Link Type	Description	Primary Purpose
None	Connects an input Item node to an output Schema node or connects an input Schema node to an output Item node	Marks Item nodes that are potential contexts or results of the Schema node's execution and maintains the statistics necessary to determine whether appropriate spin-offs of the Schema node should occur

With the exception of action and goal link types, a link must connect an item node to a schema node.

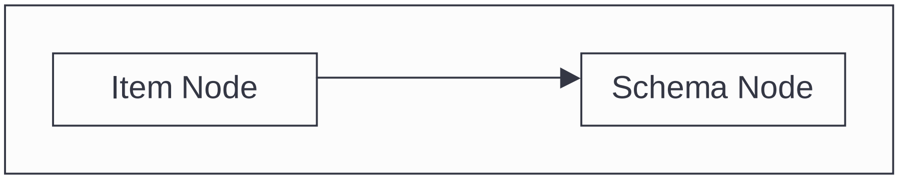

Figure 3: Links generally connect item nodes with schema nodes

A link with an item node as its input will have a schema node as its output, and is a context type link or is considered a candidate to become one. Conversely, a link with a schema node as its input will have an item node as its output and is either a result type link or is considered a candidate to become one, or is a host link that connects a learned item node to its host schema node. An action type link connects a schema node to its own action or connects a composite action to its goal node. Goal type links connect goal nodes to the item nodes that make up its goal set. Links that do not have a type (technically of type *none*) are either part of a schema node's extended contexts or extended results. These amass statistical information but do not participate in passing of activation or resistance. Figure 5 shows the early stages of a schema network running in the example environment described previously. The light colored links shown going from schema nodes to sense (item) nodes are of type *none*. The dark links from schema nodes to action nodes are of type *action*.

The statistics maintained by the links are the primary source for learning new network nodes. The first statistic of note is the relevance of the link, more technically known as the ratio of the positive and negative transition correlations. A transition correlation is the likelihood that a given item node will change its state to *on* (positive) or *off* (negative) following the schema node's activation. If either the positive or the negative-transition correlation for an item node becomes sufficiently greater than the other, then a new schema node is created with that item node linked via a link with its inclusion state matching the greater

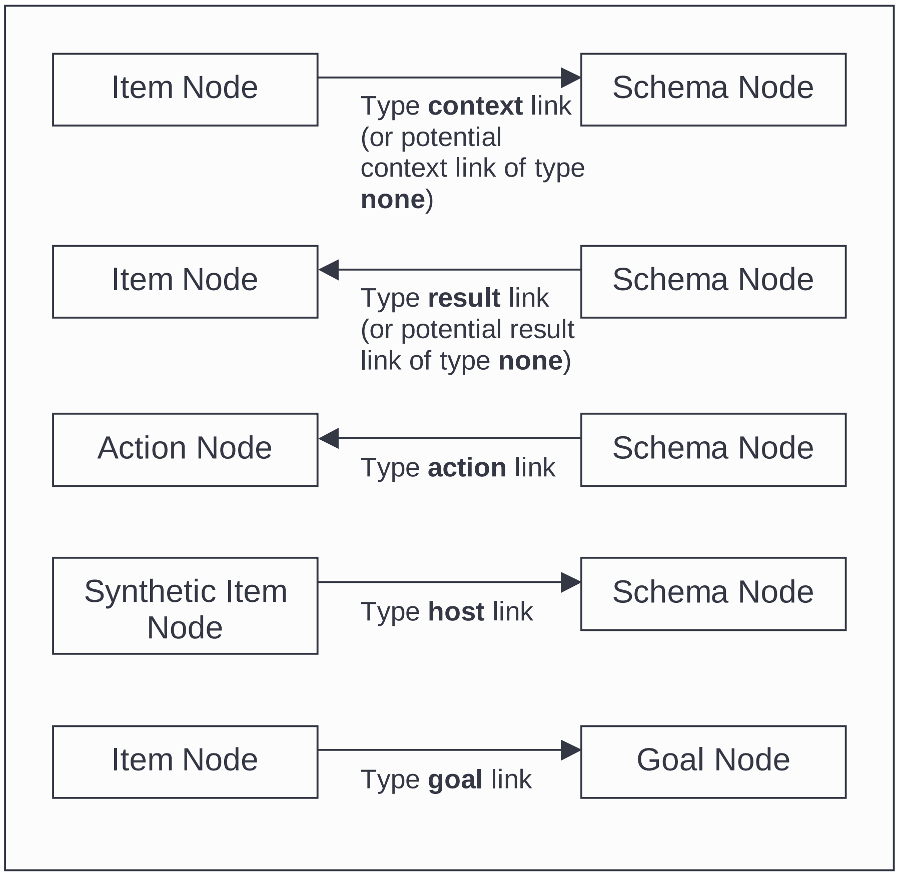

Figure 4: Possible link configurations

transition correlation. The relevance calculation is slightly different for possible result links (from schema node to item node) and possible context links (from item node to schema node). For possible context links, the correlations do not take into account the transition of states but, rather, the state that the item node was in when the schema node's action was executed. In other words, the relevance of a link measures how important the item node is to the successful completion of the schema node's action either from a context or result stand point.

A link's basic duty is to pass activation from its input node to its output node after that activation has been modified by its weight. The weight of a link is only meaningful for *result* and *context* type links, since they are the only types that transmit activation or resistance. A result link's weight is the product of its relevance and its input schema node's reliability value. For a context link, the weight is the product of its output schema node's correlation value and its output schema node's reliability. The weight can,

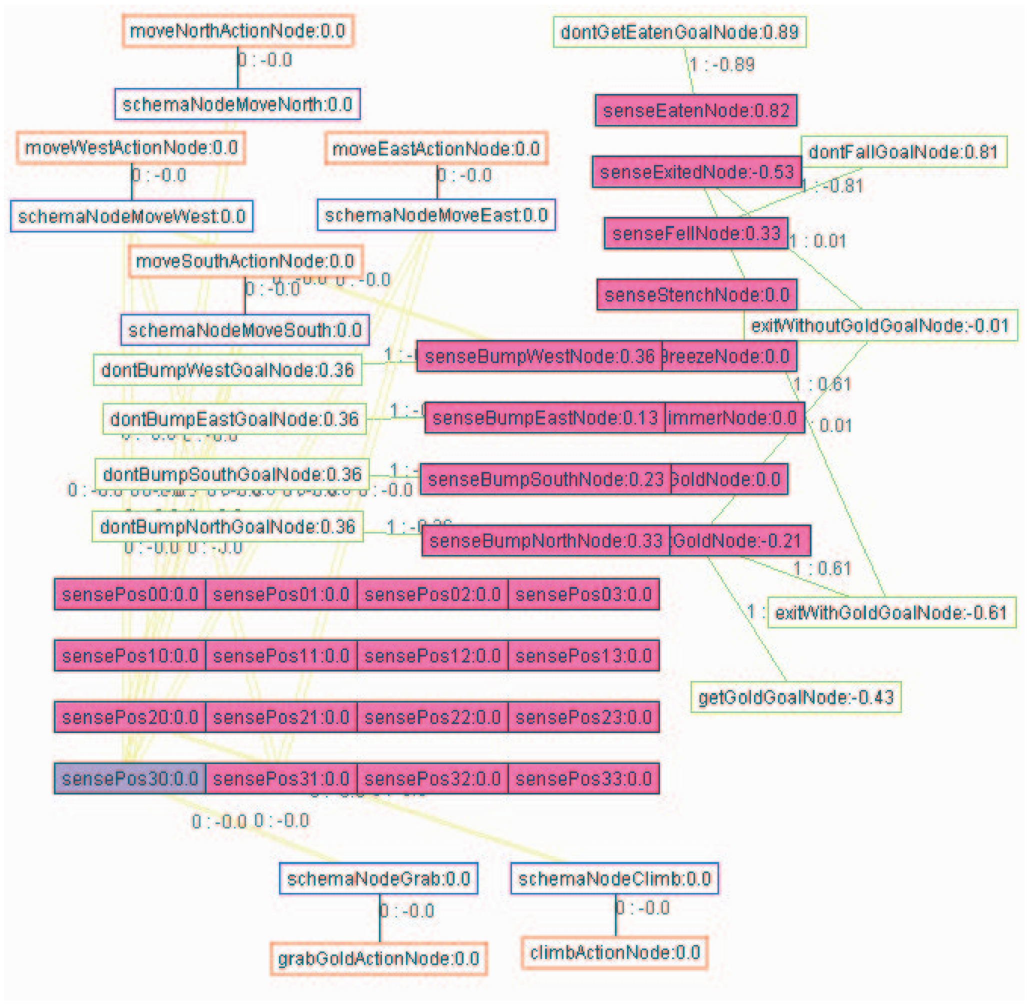

Figure 5: Early stages of a neural schema network. Light colored links are potential context or result links.

however, be temporarily modified by the resistance flowing backwards through the link. Resistance will be explained in more detail later, but for now one could think of it as analogous to the resistance in an electric circuit. It effects the activation flowing through the system by modifying the strength of the weights on the links. Resistance will also be used to learn the correct values for any intermediate goal states.

At first, a neural schema network has almost no links. In fact, at the time of a network's initialization, the only links that exist are action links from primitive action nodes to their corresponding bare schema nodes or, possibly, from primitive goal nodes to their goal set's item nodes. Figure 1, introduced earlier, shows the initial state of the neural schema network before any processing has occurred.

An attention mechanism will be discussed later that creates the initial links between nodes, such as those shown in Figure 5.

Nodes

The node, as with most connectionist systems, is one of the two main components, links being the other. A node in the neural schema mechanism can come in a variety of types that act differently based on which type it is. However, one commonality for all of the nodes in the system is that they have a state that is in correlation with their activation.

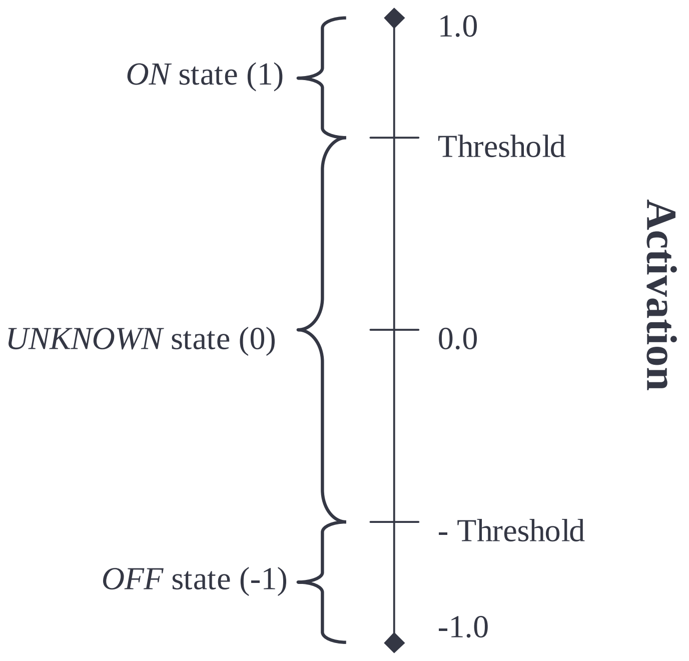

Figure 6: Activation to state mapping in nodes

Depending on the type of node, the activation may be adjusted to match the node's state, or the state may be determined by the node's activation. Most nodes have the ability to pass activation and resistance through the network, although there are limits to this ability based on the types of links that the activation or resistance could flow down. Activation may be propagated from a node when the node's activation goes over some threshold or under some negative threshold. It is said, in these cases that the node has "fired." Firing a node, however, does not mean, in the case of schema nodes, that its action is executed.

Schema nodes

The most complicated type of node is the schema node. These nodes represent the center of the explicit knowledge structures of the network. The schema node has a list of input links, which hold information about the node's context (and potential context) and a list of output links, which hold information about the node's result (and potential result). An additional output link points to the schema node's action. An individual schema node with its input context links, output result links, and action link embodies the knowledge that the item nodes pointed to by the result links will transition to a given pattern when the action node is activated under the conditions denoted by the item nodes pointed to by the context links. Put another way, the schema node states that a certain result is more likely to occur than not when the schema node's action is executed within a given context. Note that the schema node's existence does not imply that the result is likely, only that it is more likely than not when the action occurs within the context.

To measure how reliable a schema node is the node keeps a running tally of successful vs. unsuccessful activations and calculates their probabilities. The trick comes in determining when a schema node has been activated. A schema node can either be explicitly activated, meaning that it was specifically chosen for activation, or it can be implicitly activated, meaning that another schema node which shares the same action node was chosen (Figure 7). What's more, some of the statistics necessary (i.e. reliability) do not care whether the activation was explicit or implicit, while others (i.e. correlation) do need to know the difference. While the reliability measures how essential this particular schema node is in bringing about the designated result when its context holds, the correlation of a schema node measures the relationship between the action and the results. For this, the node needs to know not only when its result occurs after its action's activation, but also when it was not activated and the result occurred anyway. This is handled by letting the action node propagate its completion state (either *successful* or *failed*) down its input links to the schema nodes that could have activated it. Each schema node notes when it has been specifically chosen for activation. Therefore, the node that actually instigated the action knows who it is and can adjust its

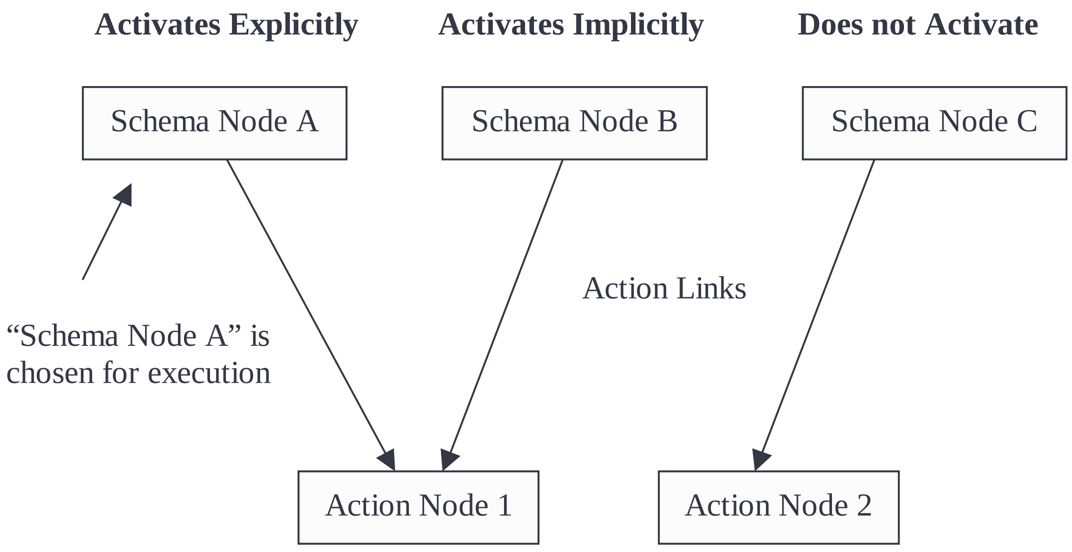

Figure 7: Explicit vs. Implicit activation

statistics accordingly; others can correctly assume implicit activation. The schema node is also watching it's result item nodes in order to note when a result transitioned without activation of either kind.

The reliability of the schema is calculated at the point when its action propagates a new state. By keeping up with the number of times that this propagation occurs (number of activations) and the co-occurrence of these activations with a successful result state, the schema node can calculate its general reliability. At the same time, the node is also keeping track of when activation follows within a short time of another successful activation. These events are included in the statistics for local reliability. Local reliability is based on the notion that things in the world tend to stay put, at least for a while. Therefore, a schema node can be generally unreliable but still locally reliable meaning that if the schema node has just been successfully activated then a repeated activation will likely also be successful.

A more difficult statistic to maintain is a schema node's correlation. Once again, correlation is defined as the ratio of the probability that the schema node's result will obtain when activated to the probability that the result will obtain when not activated. Here, since activation implies applicability, the node needs to know not only when it was activated, but also when it was *not* activated when it could have been. For this reason, all schema nodes that were not explicitly chosen for activation must check whether their actions are currently running. If they did not receive a completed action state, then their action must either be off or still in the process of running. Once the schema node knows the state of its action, it can

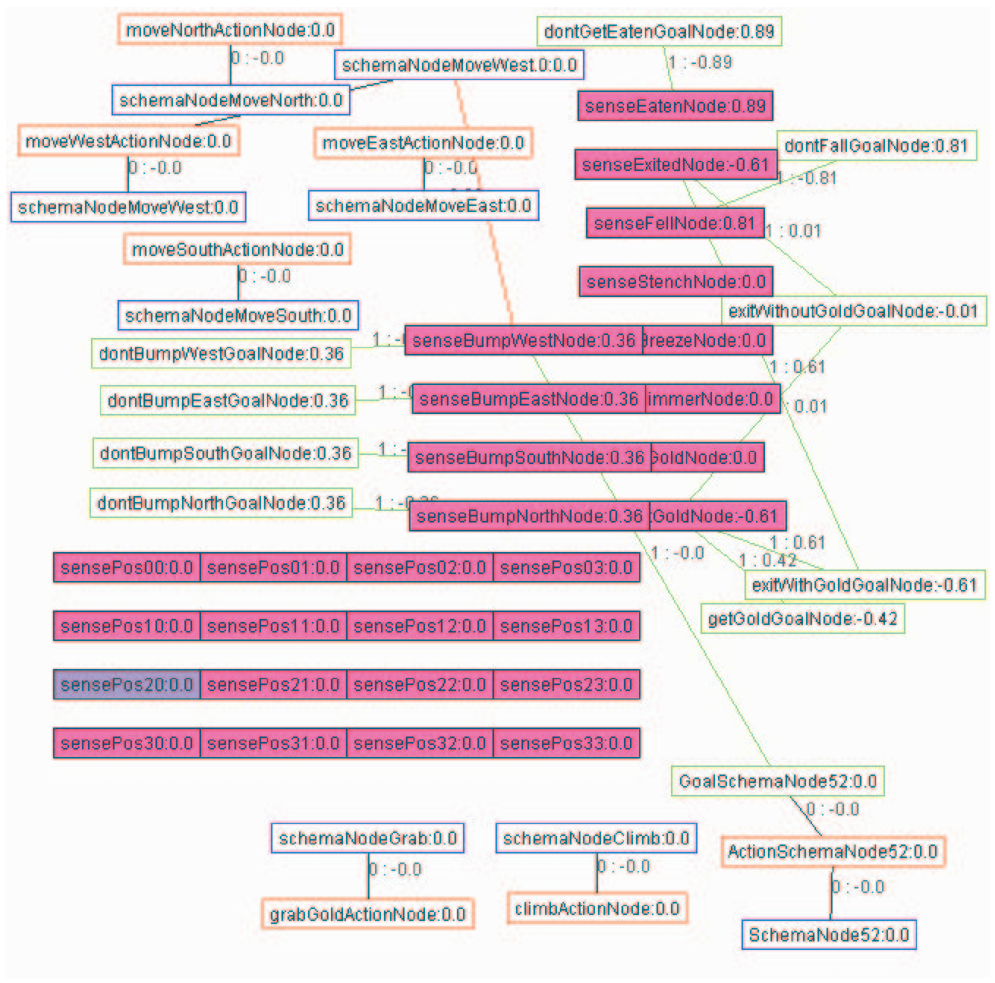

Figure 8: A neural schema network after a result spin-off. Type *none* links have been removed for clarity.

correctly calculate or defer calculation of the probability that its result state will obtain when it is applicable but not activated. For values where the negative-transition correlation probability is non-zero, the function results in mapping the raw correlation ratio to a value in [-1, 1]. Any raw value above one means that the positive-transition correlation probability is higher than its negative-transition counterpart and will be mapped to a positive value correlation which increases as the difference between the two probabilities increase. A raw ratio value between 0 and 1 means that the negative-transition correlation probability is higher and will, similarly, be mapped to a negative correlation value. In cases where the negative-transition correlation probability is zero, the correlation is equal to the reliability.

All of the statistics maintained by schema nodes in the mechanism serve one purpose: to aid in determining which action to perform at any given moment. Action selection is conducted based on the product of a schema node's activation and correlation. As we will see, many other factors come into play in determining a node's activation, but the final selection is performed solely based on these two features. The action selection process will be described in greater detail once all of the required elements have been explained.

The other main function of schema nodes is to create any new nodes that the mechanism has deemed appropriate. Even though it is, technically, the mechanism as a whole that gives the go-ahead to the creation of new nodes, it is the individual schema node that decides whether its local conditions warrant beginning the creation process. Up to this point, however, we have not talked about the initial state of the original schema nodes as seen in Figure 1. A schema node's most basic state is when it does not have any result or context links and has only its one action link; these nodes are called "bare" schema nodes. It is from this original bare state that a schema node can spawn new schema nodes with additional context or result items and can create synthetic (learned) item nodes. In instances where the result list of a spin-off schema node is new to the system, a schema node can also create a new composite action with a new bare schema node to activate that action. Figure 8 shows our example environment at the point where the hunter has discovered that moving west ("moveWestActionNode") can result in his sensing a bump to the west ("senseBumpWestNode"). This learning manifests itself in the mechanism as a new schema node that has been created with "moveWestActionNode" as its action and "senseBumpWestNode" as its result. Also, since this is a new result set for the network, a new action node is created along with a corresponding goal node and bare schema node that can be seen to the bottom right of the figure. The goal node is connected to "senseBumpWestNode" denoting the new result set as the goal of "ActionSchemaNode52".

As with any other type of node, a schema node can pass its activation when it fires. This is distinctly different from the mechanism choosing a schema node to activate. Remember that activating a schema means that the schema node is applicable and that the mechanism has chosen to perform its action. Firing, on the other hand, occurs whenever a schema node is applicable and its activation goes over a threshold or under a negative threshold. When this happens, it sends activation through its result output links.

Item nodes

An item node corresponds to some state in the environment. Primitive item nodes are connected directly to some sensory apparatus. Unlike schema nodes that must rely on other parts of the mechanism to provide them with their activation, primitive item nodes set their own activation based on the values input from the sensory apparatus. For switch-like senses, such as a bump sensor, an item node's activation might be set to 1 if the bumper is depressed, or –1 if the bumper is not depressed. Other senses, such as proximity sensors, might map an item node's activation to a continuous value between –1 and 1. It is important that each primitive item node's range of activation be in [-1, 1]. A threshold function determines if an item node's state is *on*, *off*, or *unknown* based on the activation of the node. If a node's activation range were only in [0, 1], for example, then the threshold function would set the node's state to *on* or *unknown* but never *off*. For learning purposes, such a situation would yield incorrect or, at best, unreliable information about the node's relation to schemas and actions. In our Wumpus World environment, the state of each of the hunter's senses is boolean, rendered in the network as [1] for *on*, and [-1] for *off*.

When a schema node is not generally reliable but its local reliability is high, then it is a candidate for synthetic item creation. Unlike primitive item nodes, learned item nodes are not directly connected to any sensory devices. Instead, they are connected via a "host" link to the schema node that spawned them and any of its subsequent context spin-offs. Learned items nodes are an important part of the discovery process within the mechanism. They establish the counter-factual assertion that a given action *would* produce a given result if that action were taken under a specific set of circumstances. An example might help to show the importance of this element. Suppose that an agent is in a dark room. The agent has noticed that, sometimes, when it takes a step forward, that it runs into something. In general, the act of taking a step forward does not usually result in the agent running into an object. However, if it has just tried to step forward and stubbed its toe, and then repeats the same action, it will stub its toe again. A learned item is created with this schema as its host. The schema is said to reify the item. In other words, the schema designates the conditions under which the item's state is *on* (or *off*) even though the agent doesn't initially know what those conditions are. Further spin-offs of the schema node will refine its ability to note the state of this learned item node. The mechanism can treat that item as an object such that it does not need to actually move forward while at the special location, but can create chains of schema nodes that move the agent around the object without running into it. Even though this is a simple example, it demonstrates how learned item formation can change the behavior of the system in a very important way. From this point, schema nodes can be created whose contexts or results contain the newly created item node.

Unlike primitive item nodes that rely on sensory apparatus to determine their activation and state, learned item nodes must find some other means to determine their state. There are three methods that a learned item uses to maintain its correct state. The most obvious is the successful execution of any of its host schema nodes. Since the learned item node defines the conditions under which the host schema nodes will be successful, a successful execution of one of those nodes is the clearest indicator of the learned item's state. Secondly, if any of the learned item node's host schema nodes are applicable (their context set is satisfied) then the item node should be turned on. All of the host schema nodes will have the same action and result; therefore, context spin-offs serve to more accurately specify the conditions in which the result follows the action. Having any of a learned item node's host schema nodes have their context satisfied is equivalent to stating that the conditions are correct for the result of the schema node's action to obtain and, hence, the learned item node is turned on. The final way that a learned item maintains its state is through the prediction of that state by being included in the result set of some schema node.

The learned item reifies its hosting schema nodes by allowing the mechanism to treat the arbitrarily abstract conditions laid out by the host's context as a real and palpable entity. In the case where one of the learned item node's host schema node's context is satisfied, and, therefore, the item node's state is *on*, the mechanism does not have to perform the action to know (or at least guess) that the result would occur. Under these circumstances the mechanism could learn to perform actions based on the existence of the conditions represented by the learned item node without having direct sensory manifestation of the result. The mechanism is thereby treating the learned item node as though it represented the definition of the existence of the object.

It is important, however, not to confuse reification with definition. In our stubbing of the toe example, a learned item node might be created that reifies the schema node that states that (after one or more context spin-offs) at a specific location the action of moving forward results in stubbing the toe. One might erroneously conclude that the learned item defines the existence of the malicious object since the mechanism treats that item as though it represents the object. Instead, what the item represents is the conditions under which the existence of that object (or some other object that the agent might hit its foot on) could be confirmed. Over time, the agent can create context spin-offs of host schema nodes that continue to more closely specify the reification conditions, perhaps to a degree that is near infallible. Even so, the agent will never be able to construct a *definition* of the object, only closer approximations of confirmation conditions.

The importance of this point comes out when one examines arguments against artificial intelligence being able to construct truly novel concepts (Fodor 1975). The argument, put much too simply, states that an artificial system can only create reorganizations and abstractions of its inputs and, therefore, can never create concepts for things that are incomputable or for which only incomplete data is available. The logic of the argument is absolutely correct if one assumes that a definition is the infallible concept of something. The idea of learned items in the schema mechanism gets around this argument by not trying to define objects or concepts. A counter argument, of which the schema mechanism is an example, is that definitions of the type called for in the argument do not exist even in humans. For an interesting exercise to demonstrate this point get a small group of people together and try to come up with an infallible definition for a chair. You will, most likely, start by listing the properties of a chair and will quickly discover that for every property there is a counter example. You may then resort to defining the chair by what one does with it, namely sitting. But one can sit on a lawn, or a rock, or a table. Finally, you may come up with a sort of fuzzy definition saying that everything has some degree of chairness, even a lawn is a chair if one sits on it. Even for something as basic as a chair, strict definition is difficult if not impossible. There are, of course, strict definitions for such things as mathematical concepts, a square, for example. Even so, it is quite possible that humans maintain only an approximation of the meaning of the elements that make up even these definitions. The schema mechanism learns, not fuzzy sets, but fuzzy conditions that allow the agent to interact appropriately in every way with that object or concept.

Action nodes

Action nodes perform the function of actions in the network. Although the action nodes are not the actions themselves within the implementation, this point is a technicality and not important to how the mechanism functions. Similar to item nodes, there are primitive and learned action nodes. For a primitive action node, an action controller is provided that directly connects to some actuator. These would be such things as particular muscle movements or turning on an engine.

Learned actions are created whenever a schema node creates a new schema node with a novel result. Here, “novel” is used to mean a result set that is not shared by any other schema node. When this occurs, the creating schema node carries out three additional functions, creating a new action node, creating a new goal node, and creating the bare schema node that allows the system to learn about the results and contexts of executing the new action. An example can be seen in Figure 8 where “SchemaNode52,” “ActionSchemaNode52,” and “GoalSchemaNode52” have been created in response to the new result represented by schemaNodeMoveWest.0. The action controllers for learned actions in the neural schema mechanism are very simple and perform the same task for every composite action. Essentially, their only job is to monitor the state of their goal node and to activate the goal node when that action is executed. This activity will make more sense once the details of goal nodes are explained. The purpose for this multi-layered approach to action execution is merely to allow a single action node class in the implementation to provide the interface for all possible actions.

Goal nodes

Goal nodes, essentially, have one main purpose; that is to provide value judgments to specific states of the environment or to individual item nodes; they are the source of motivation in the network. The original schema mechanism accomplished this task through the use of primitive, instrumental, and delegated importance values associated with items. One difference in the new implementation is that goals can be primitive[4]. In the original schema mechanism, a goal was always associated with a composite action; this is not the case with the neural schema mechanism. A primitive goal node is analogous to basic drives; its purpose being to influence the mechanism in such a way as to bring about its goal state, however that might

[4] Drescher describes the possibility of moving the innate drive for discovery, which was built into the mechanism as a pseudo-randomly changing focus of schema selection, to an item that would register whether significant learning was going on. Goals, however, were only internal structures of composite actions.

be accomplished. Whether the goal node is primitive or learned, the method for bringing about its goal state is the same, namely the spreading of resistance[5] through the network.

A primitive goal node in the new mechanism will have an intrinsic primitive resistance value provided at design time. Instrumental value, on the other hand, is imparted through the propagation of that resistance at runtime. Delegated resistance value accrues in learned goal nodes. Remember that new goal nodes are created whenever a schema node is created whose result is new to the network. For this reason, there will be a goal node associated with every unique set of item nodes that is referenced as a result of any schema node. The delegated resistance is calculated as follows. First, the maximum amount of resistance flowing through all applicable schema nodes is determined. This step measures the greatest benefit that the agent believes could be reached in one time step from the mechanism's current state. Each learned goal node then receives that value and averages it with previous values noting whether the goal node's state is currently *on* or *off*. For example, if a goal node is manifest, meaning that all of the item nodes' states to which it is linked match their corresponding goal link's include state on that cycle, then the new value just received would be averaged with the previous values received while in the *on* state. The goal node's delegated resistance value is then calculated as the average maximum resistance when the goal is met (*on*) minus the average maximum resistance when the goal is not met (*off*).

Delegated resistance value in learned goal nodes is the method by which the mechanism learns the desirability or non-desirability of intermediate states or sub-goals. The item nodes and their states referenced by a goal node and its goal links make up a possible intermediate state along the path to one or more primitive goals. Determining the relative value of attaining (or avoiding) that state irregardless of the current drives of the system will allow the mechanism to anticipate states that may be useful in the future. The mechanism then generally attempts to maintain useful states or avoid detrimental states as long as there are no stronger conflicting goals that require otherwise. In other words, the system begins to pursue states for their own right without regard to the ultimate drives that they serve. This ability to pursue (or avoid) goals that are in service of some lower-level drive without reference to that drive is common in humans – even to the point of having mistaken priorities. Even so, most humans do quite well satisfying their basic

[5] Although the metaphor of electrical resistance is used, we will actually positive resistance to increase activation flow and negative resistance to decrease activation flow. This has proved to be less confusing.

drives. Although the neural schema mechanism has not produced behavior tendencies toward these human extremes, the theoretical possibility of this feature of goal nodes is intriguing.

In implementation, goal nodes perform two main functions, to compute a state function based on its goal links and to send resistance (positive or negative) backwards through the network when its action is executed. The goal links are approximately equivalent to a context. When all of the states of the item nodes pointed to by these links obtain, then the goal node's state should be *on*, otherwise its state should be *off*. It might be possible to derive useful inferences and actions by letting the goal node's activation represent the degree to which its state obtains. Although in the current implementation this is not done, it would be a trivial task to modify the system to function in this way.

A goal node will periodically poll its goal links to determine its state, which is then passed on, in the case of learned goal nodes, to the action that has this as its goal. When the goal node's state turns *on*, this is noted by the goal node's action node, which will then propagate an action-completed signal to all of the schema nodes that might trigger it. Since obtaining the goal node's state is the defined purpose of the learned action node, having the goal node's state obtain without the explicit execution of the action node is equivalent to the implicit execution of the action. Propagating the action-completed signal whenever the goal state manifests lets the schema nodes connected to the action node learn about contexts and results of composite actions using both implicit and explicit activation just as they would with primitive actions.

Resistance, activation, and action selection

The ability for the neural schema mechanism to pursue multiple conflicting or complimentary goals is accomplished via the propagation of resistance through the network. Resistance begins in a goal node and can be positive or negative. Even though we have borrowed the electricity metaphor for resistance the values are reversed to decrease confusion. We would say that a goal that is desired would propagate a positive resistance and one for which avoidance is preferred would propagate a negative resistance. Figure 9 shows a portion of the early state of a neural schema network for the Wumpus World environment. The colored in nodes in the figure are some of the primitive item nodes for the environment and the outlined nodes are the primitive goal nodes associated with them. The numbers at the end of each node name displays the resistance of the node. Each link has two numbers labeling it; the first is the include state

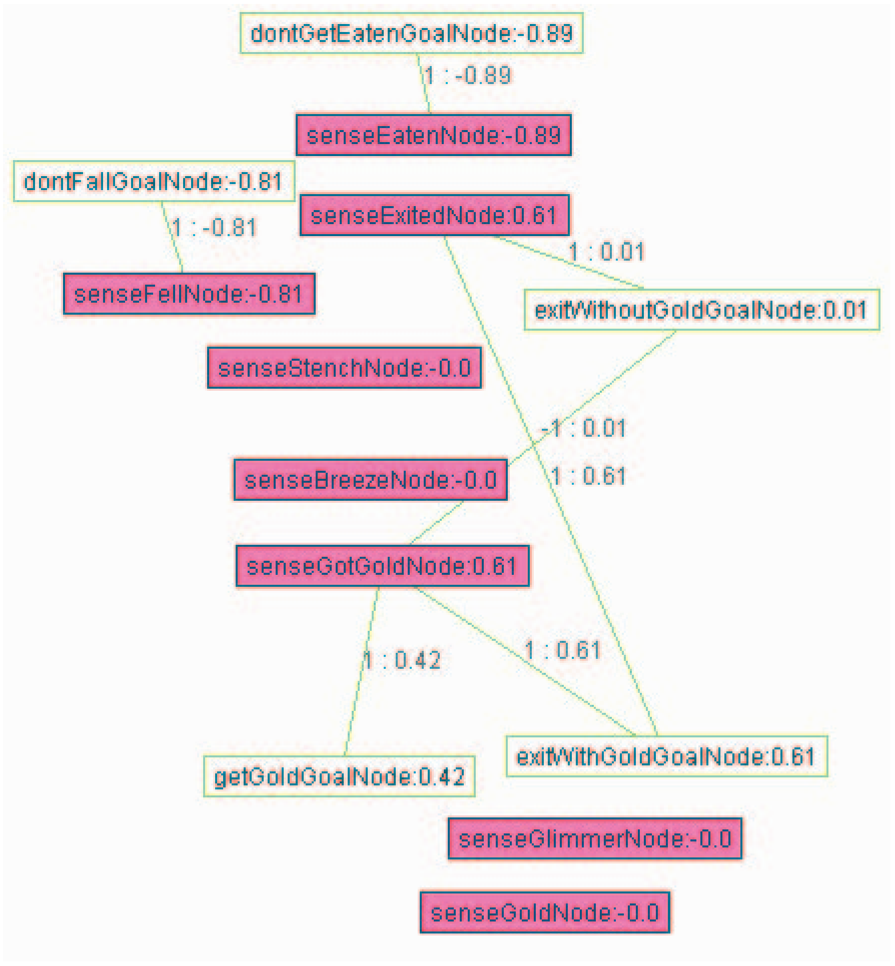

Figure 9: Some of the sense (item) nodes for the Wumpus World environment and their primitive goal nodes. The number at the end of the node names is the resistance at that node. The first number associated with a link is its include state and the second number is the resistance flowing across the link.

(either –1 or 1 in this case) while the second is the resistance flowing over that link. The desirability or undesirability of any given state of an item node with respect to a particular goal node is defined by the resistance of the goal node and the include state. In the figure, for example, the "dontFallGoalNode" has a resistance value of –0.81. This alone tells us that whatever state is represented by that goal node's goal links is a state that the agent wishes to avoid (although there could be other goal nodes that want to achieve this same state). When we also look at the goal link connecting the "dontFallGoalNode" with the "senseFellNode" item node, which has a positive 1 as its include state indicating an *on* state, we can determine that the "dontFallGoalNode" is meant to avoid turning *on* the "senseFellNode" item node. Another element of Figure 9 that is important to note is the fact that not all primitive item nodes must be linked to some primitive goal node, “senseStenchNode” for instance. In the initial state of the network, there is nothing to tell the agent that the stench percept is one to be avoided; it is up to the agent to decide for itself over time if this sense is one to be sought or avoided.

Just as with activation, there are limits as to which links are permitted to transmit resistance. Item nodes can only send resistance through result links to schema nodes and schema nodes are only allowed to send resistance through context links to item nodes. This process maintains the path to the instigating goal node with stronger resistance values occurring closer to that goal.

Resistance does not actually change the activation of a node; instead it affects the amount of activation that passes through a link. The best way to explain this would be to say that the resistance in the system alters the contours of the network. One can think of the neural schema network without resistance as occupying a flat plain within a three-dimensional space. Activation is like water trying to progress from an existing state to a goal state with the weights of links providing some hills and valleys that hinder or assist the water in its journey. The network landscape at this point in our analogy represents a purely logical perspective of how to get from one state to another. When resistance is added, the contours of the network change to reflect a combination of logic and desires. What once was a valley might now be a flat plain or a veritable gorge; in the same manner, a hill might become a mountain or a plateau.

Goal nodes, the source of resistance, are activated in one of two ways. If the goal node is attached to a composite action then the action node can activate the goal node. If, on the other hand, the goal node is primitive, it might have some direct attachment to a proprioceptive sensor that would register something internal to the agent such as hunger. In a circumstance such as this, the goal node would be attached to a primitive item node or nodes that represent the proprioceptive state. While this may seem redundant, it is not. The item nodes, themselves, only register a state of the environment and do not make judgments as to its desirability or undesirability. Goal nodes, however, have the explicit purpose of deciding which environmental states should be sought and which should be avoided. In our example, as displayed in Figure 9, FlagSenseNode2 is currently *off* and, therefore, in the desired state. For this reason there is no resistance being propagated to it from goalNode2 or from it to schemaNode2 or schemaNode4. FlagSenseNode1, on the other hand, is also currently *off*, which is not the desired state. In this case goalNode1 is propagating resistance to it and it is in turn propagating resistance to all of the schema nodes that have it as a result. The mechanism has learned that performing ActionNode1 will result in FlagSenseNode1 turning *on* and represented this knowledge with the creation of schemaNode1.0. A positive resistance is sent to schemaNode1.0 in an attempt to increase the activation that might get passed to it and, consequently, increase the likelihood that schemaNode1.0 will be activated. The mechanism has also learned that performing ActionNode3 will turn FlagSenseNode1 *off* and represented this knowledge by creating schemaNode3.0. When resistance is sent to schemaNode3.0 it is converted to a negative value that can be propagated in an attempt to inhibit activation from coming to the node and, thereby, decreasing the likelihood that schemaNode3.0 will be activated.

For goal nodes in service of composite (learned) actions, resistance is only propagated for a short period of time (based on the expected duration of the action) unless the action is repeatedly selected. Primitive goals will continuously spread resistance based on the current state of the system. This allows a primitive goal, such as hunger, to be pursued to varying degrees at all times. When the goal node's need is high, the agent may pursue it single-mindedly; however, once such a goal has been satisfied, it would be reduced to a level that has little to no effect on the behavior of the system.

While resistance begins at the item nodes pointed to by active goal nodes, activation has its source in item nodes that represent the current state of the environment. Activation flows in an opposite manner to resistance. An item node can pass activation through its context links to schema nodes, which can pass activation through their result links to item nodes. As mentioned previously, activation can be transmitted from any node whose activation level exceeds a threshold or drops below a negative threshold. Since activation is spread through the system by numerous nodes, there must either be a limit to how high the total activation of a system can grow or the decay rate must be tuned appropriately to manage this level. Since the number of nodes in the network is always in flux, the total number of nodes in the network must be taken into account. In the neural schema mechanism, both methods are used at different times. The total amount of activation of a network is limited to half of the total number of nodes. If the total activation of the network is below this maximum, then only a decay rate is used to reduce the activation of a given node. However, if the total activation goes above the maximum, then the activation is normalized so that each node maintains its correct percentage of the whole but the total activation of the network remains equal to the maximum. The use of half of the total number of nodes as the maximum for network activation is not arbitrary but may need to be adjusted in future experiments. It may also be possible to allow this value to change during a run. For instance, lowering the value from 0.5 to 0.25 might promote more abstract thinking by forcing the system to highly activate fewer, hopefully more general, nodes.

Even though activation plays an important role in action selection, it is not the only element to consider. In addition to activation, a schema node's applicability is taken into account. Under normal circumstances, for a schema node to activate (perform its action) the node must be applicable and have the highest activation of all applicable schema nodes. The use of resistance increases the likelihood that the schema node selected for activation lies on some path to a goal state. In addition, the fact that the resistance values become stronger the closer a node is to the goal state, means that the schema node chosen for activation is more likely to be one which is fewer steps away from achieving the goal. This method gives the neural schema mechanism the ability to be opportunistic while pursuing multiple goals.

"Consciousness" and attention

It was stated in the previous section that, under normal circumstances, a schema node must have the highest activation of all applicable schema nodes to be chosen for activation. The other way that a schema node could become active is through a response to a "conscious" broadcast. The term "conscious" is used here only to denote that the mechanism implements a portion of a psychological theory of consciousness[6]. It is neither our intent to suggest nor our belief that the mechanism displays true consciousness. That being said, it is our hope that, within the limited scope of the "conscious" mechanism implemented, one might be able to form hypotheses regarding human consciousness or be able to discover discrepancies that could be remedied in future versions of the system.

The particular theory of consciousness that has been partially modeled in the neural schema mechanism is Bernard Baars' global workspace model (1988, 1997). This theory puts forward the claim that human cognition is implemented by a multitude of relatively small, generally unconscious, special purpose processes. Within this multi-agent system, groups of such processes, called coalitions, compete for entry into a global workspace. The workspace serves to broadcast the coalition's message to all the unconscious processors, in hopes of recruiting other processors to help solve the current impasse or to take

[6] A more complete implementation of this theory of consciousness is described by Bogner (1999).

note of the current novel situation. The number of processors that can be in the workspace at any one time is limited but not set at a specific number.

Even though in humans consciousness plays an important role in solving problematic situations, it is easy to recognize from one's own experience that consciousness is not only used in these circumstances. To the contrary, humans are usually conscious at any given moment without, necessarily, being in the midst of solving some unresolved issue. This is where attention comes into play. In the theatre metaphor presented by Baars, the spotlight of attention is the method by which coalitions get into consciousness. However, the theory does not dictate what criteria should go into deciding which coalitions are attended to, nor does it dictate that all coalitions that get the spotlight of attention shined on them must be a novel situation or problem, only that novelties have an increased probability of entry.

Attention in the neural schema mechanism has a similar purpose to the one created for the original schema mechanism by Foner and Maes (1994); namely, to reduce the computational requirements of the system. Ours, however, works by using a combination of the change in a node's activation over time, the magnitude of resistance flowing through the node, and the node's association to other nodes recently in "consciousness." Each node in the network has a degree of "consciousness." In this way it can be thought of a similar to activation. All node types, with the exception of action nodes, are candidates for inclusion into "consciousness." The first two elements that make up the calculation for the degree of "consciousness" in any given node are determined without respect to the node's relationship to any other node. First, the slope is calculated for every node in the network. Slope is defined here as the absolute value of the average change in activation over the last five cycles, plus the absolute value of the resistance at the node multiplied by a resistance factor. The resistance factor is used to balance attention between highly active nodes and nodes deemed important in obtaining or avoiding goals. The resulting slope value is proportional to the rate of activation change in the previous five cycles modified by the resistance level. This portion of the equation gives a measure of the node's novelty (rate of change of activation) and usefulness in accomplishing current goals (resistance). This value is combined with the amount of "consciousness" that was sent to the node through its links. The passing of consciousness is meant to provide the agent with some continuity of thought from one cycle to the next. Under most circumstances this give the agent something akin to a train-of-thought event though highly novel or unexpected occurrences could force the agent to jump to a new awareness unrelated to the previous contents of "consciousness." At this point the "consciousness" values for all of the nodes in the system are normalized in order to maintain a constant value for the total amount of "consciousness" in the system.

The focus of attention (the number of nodes in the spotlight) is dynamic and can change based on the overall state of the system. After the "consciousness" is calculated for every node, the values are totaled and a percentage of that total is noted. The percentage of the total slope is a constant set up at the beginning of a run that dictates how inclusive the spotlight of attention is. A value of thirty percent, for example, would say that only those nodes whose degree of "consciousness" makes up the top thirty percent of the total would be in the spotlight. The percentage quota system allows the number of elements that get into the spotlight of attention to grow when the system is in a relatively stable state and narrow when the system is experiencing drastic change or strongly pursuing a particular goal. A stable state for attention occurs when activations are not changing rapidly and no one goal (or small number of goals) is being robustly pursued. For example, if the total "consciousness" of the system were spread somewhat evenly over twenty different nodes, then a thirty percent spotlight value might include five or six different nodes. If, on the other hand, just five nodes provided fifty percent of the "consciousness" in the system, then a thirty percent spotlight value would only encompass two or three nodes. As can be seen from the example, changing the spotlight percentage value effects the general focus of attention while still allowing it to adjust to changing situations. Experimentation needs to be conducted to determine an optimal value for this percentage that maximizes learning by creating enough links between nodes while minimizing the branching factor of the resulting network.

The branching factor of the network is largely determined by the creation of links between attended nodes. Once a set of nodes is selected for "consciousness" by the attention mechanism, two things happen. The first task is to update the links between all of the nodes in the spotlight. A node cannot have a link to itself, and action and item type nodes cannot link to other action and item type nodes. If two nodes, say Node-A and Node-B, had no previous connections between them, then two new links are created, one with Node-A as input and Node-B as output, and one connecting the two in the opposite direction. This is in contrast to the fully connected nature of the original mechanism. Previous research has shown that an attention mechanism that operates in a similar manner with regards to this first task, reduces the branching factor of the mechanism without significant lose of learning capabilities (Foner & Maes 1994).

The second thing that happens to "conscious" nodes is that a broadcast is sent out to all schema nodes in the network. The broadcast consists of a set of item nodes. If an item node is in the spotlight, then it is, itself, one of the item nodes broadcast. If, on the other hand, attention is broadcasting information for a schema node or goal node, then that node's context set is broadcast. For goal nodes, we would say that their goal set is broadcast. In total, the broadcast will contain all of the item nodes represented by the coalition of nodes in the spotlight. Schema nodes respond to the broadcast if any of the items transmitted correspond to any or all of its result set. The schema node responding to a "conscious" broadcast sets an internal flag that denotes that it is responding and increases its activation proportional to the percentage of its result set represented in the broadcast. Responding to a broadcast does not automatically guarantee selection for activation, but it does increase the likelihood. The flag denoting a node's response to the broadcast is an overriding factor to the applicability rule for action selection. It is in this way that an agent can discover new paths to a solution or goal state that did not previously exist or were not active highly enough to become chosen for activation. Selecting schema nodes whose result sets match the desired goal under these special circumstances even when the context is different or unknown allows the agent the possibility of quickly learning the correct context or alternate context for achieving the goal. For example, if the schema node chosen for activation by a response to the broadcast has no context at that moment then its selection serves to decrease the time necessary to discover the correct context due to its increased likelihood of activation. On the other hand, if the chosen schema node does have a context and the action turns out to be successful in bringing about the expected result, then the mechanism has the possibility of creating, in very few steps, a new schema node with a complex context solution to arriving at the broadcast goal. Otherwise the system would have had to build up the solution from a series of schema node creations.

The general formula for attention serves a number of different purposes. First off, it increases the likelihood that unexpected occurrences will become "conscious". Because of the way activation is spread within the system, the mechanism displays the psychological phenomenon of priming and, along with the "conscious" broadcast, the phenomenon of habituation. Priming is a well-studied effect in the human brain where the future activations of concepts can be influenced by similar currently active concepts. The classic example of this effect is the disambiguation of a word with multiple senses. In the stereotypical version of these experiments, a subject is shown the word “stream” and is then shown the word “bank” and is asked to provide the meaning for “bank.” In this situation, the subject is likely to give the sense of the word “bank” relating to “river bank”. Another subject first shown the word “money,” however, is likely to respond with the sense of “bank” relating to a financial institution. The notion is that concepts related to the first word presented receive some activation, which influences future activations toward those related concepts. It is likely that unconscious expectations are manifestations of this same process. When a person performs an action that they have performed numerous times in the past, such as turning on a light in their house, activation flows to sensory nodes that should feel the light switch at the next moment. When that state of the environment obtains and those senses are triggered, the activation of the sensory nodes rises only slightly; the expectation has been confirmed and the action completes unconsciously. It is only when something unforeseen occurs that a sensory node must jump from a relatively low activation to a high activation (or vise versa). This unexpected change in activation is the first part of what the attention mechanism uses to decide what becomes conscious.

Habituation occurs as a necessary precursor to reliable priming. Before the links between two nodes can become strong enough to prime a behavior unconsciously, the links will have had to have been used a number of times, most likely, consciously. Repeated successful use of a given chain of schema nodes increases the weight of the links between them, causing a chain, that initially drew the spotlight of attention, to prime itself in such a way as to minimize the change in activation between the selections of schema nodes in the chain. At the point where all of the schema nodes in the chain can be activated in sequence without drawing the spotlight of attention, the chain has been habituated. While it is probable that schema nodes, which activate in sequence but are weakly connected (through their result/context item nodes) will be “conscious” due to their changing activation slope, it is not assured. Enforcement of this policy would better fit the psychological theory but is not currently implemented.

The other element in the formula for attention is the resistance of a node. As shown previously, resistance is the measure of how much a given node is in service of the agent’s currently desired goal context (or contexts). For this reason, the addition of resistance to the formula increases the degree to which the agent's current goals are represented in "consciousness."

The use of "consciousness" does not always change the behavior of the system. To the contrary, for most situations the mechanism would have chosen the same set of schema nodes in the same order due to the other factors already laid out. At times such as these, one could say that the agent is "consciously" aware of what it is doing or experiencing, but that "consciousness" plays no discernable role in the agent's behavior. However, under circumstances where a path is not clear or where two (or more) separate chains of schema nodes must converge on a goal state, "consciousness" provides a solution.

Conclusion

Neural schema mechanism is an attempt at creating an autonomous agent control structure such that an agent using it would perform well even when placed in an environment for which it has little or no prior knowledge. From an initial state consisting only of a set of actions, sensory inputs, and primitive goals/drives a neural schema agent can discover what its actions do and how to achieve its goals. It learns about its environment through its own experience and makes value judgments based on its own internal drives. There are still several avenues of experimentation and extension that would be of benefit. An episodic memory, for instance, might be of use. This could be accomplished by allowing a subset of the item nodes to trigger recall from a memory store of past similar states and be read into a separate set of item nodes. An addition to this might even limit the recall triggers to those item nodes currently in "consciousness."

There are two drawbacks to the neural schema mechanism in its current state. First, the learning rate of the system with regard to the rate of new knowledge create is slower than had been hoped and may well be slower than other methods although direct comparison is difficult and has not been attempted as of yet. The second drawback is the computational complexity of the system. The network within a run does not grow exponentially fast due to the utilization of several methods that suppress redundant or obviously fruitless paths and due to the use of trash collecting techniques that remove unnecessary nodes after they have been deemed erroneous or very rarely used. Even so, the system does display a small but noticeable slowdown on a PC with a 933Mhz processor and 500MB of RAM when the network begins to get large (around 750 nodes). The mechanism has not yet been run to its breaking point but the question of scalability is a serious issue. If a relatively small environment such as the Wumpus World can tax the system computationally, then it is unclear if larger environments could be handled in real-time. There is always the possibility that new hardware will render this issue moot, but optimizations should, nonetheless, be explored.

Despite these issues neural schema mechanism has shown promise from both an artificial intelligence and cognitive science perspective. In essence, a neural schema agent, like a human child, creates its own reality. It takes agent autonomy one step further, allowing an agent to not only decide on the path to its goal, but also to create new goals. In addition to being inspired by human psychology, neural schemas demonstrate clear advantages to the use of those cognitive phenomenons. The way in which consciousness, in particular, has been modeled allows the mechanism to display behavioral characteristics similar to what the psychological theory describes as benefits of consciousness (Baars 1988, 1997). The mechanism has shown the ability to choose appropriate or reasonable actions more often than randomness would dictate in situations where complete knowledge of the world in that state had not yet been reached. In addition, while exact statistics have not yet been gathered with regard to the learning rate increase expected with "consciousness," the difference does seem to be noticeable by an observer.

The neural schema mechanism's motivational system has shown that it has the ability to not only pursue multiple goals, but to do so in such a way as to reduce its own risk. For example, in the Wumpus World environment the hunter agent will often avoid squares that trigger the "stench" perception (those squares adjacent to a wumpus) even though the "stench" percept was not one that had been attached to an innate avoidance goal. No external entity had to tell the hunter agent that a "stench" represented something bad for it; the agent simply discovered it on its own.

References

Baars, B. (1988). *A Cognitive Theory of Consciousness*, Cambridge, MA: Cambridge University Press.

Baars, B. (1997). *In the Theater of Consciousness,* Oxford, UK: Oxford University Press.

Bogner, M. (1999). *Realizing 'consciousness' in software agents*. Unpublished doctoral dissertation, The University of Memphis, Memphis, TN.

Drescher, G. (1985). *The Schema Mechanism: A Conception of Constructivist Intelligence*. MIT M.S. thesis.

Drescher, G. (1987). A mechanism for early Piagetian learning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*. AAAI-1987.

Drescher, G. (1991). *Made Up Minds: A Constructivist Approach to Artificial Intelligence*, Cambridge, MA: MIT Press.

Fodor, J. (1975). *The Language of Thought*, Cambridge, MA: Harvard University Press.

Foner, L. & Maes, P. (1994, August). Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning. In *Proceedings of the Third International Conference on Simulation of Adaptive Behaviour '94*. Cambridge, MA: MIT Press.

Franklin, S. (1995). *Artificial Minds.* Cambridge, MA: MIT Press.

Maes, P. (1989). How to Do the Right Thing. *Connection Science Journal*, *1(3)*, 291-323.

Maes, P. (1992). Learning Behaviour Networks from Experience. In F. J. Varela & P. Bourgine (Eds.), *Toward a Practice of Autonomous Systems*, *Proceedings of the First European Conference on Artificial Life*. MIT Press/Bradford Books.

Ramamurthy, U., Bogner, M. & Franklin, S. (1998). Conscious Learning In An Adaptive Software Agent. In *Proceedings of The Second Asia Pacific Conference on Simulated Evolution and Learning*. (pp. 24-27). Canberra, Australia.

Russell, Stuart and Norvig, Peter (1995). *Artificial Intelligence: A Modern Approach*, Upper Saddle River, NJ: Prentice-Hall, Inc.