Learning and decision-making in artificial animals

Citation for the original published paper (version of record):
Strannegård, C., Svangård, N., Lindström, D. et al (2018). Learning and decision-making in artificial animals. Journal of Artificial General Intelligence, 9(1): 55-82.
http://dx.doi.org/10.2478/jagi-2018-0002

Journal of Artificial General Intelligence 9(1) 55-82, 2018
DOI: 10.2478/jagi-2018-0002

Submitted 2017-05-16
Accepted 2018-06-13

Learning and Decision-making in Artificial Animals

Claes Strannegård CLAES.STRANNEGARD@CHALMERS.SE
Department of Computer Science and Engineering, Chalmers University of Technology, Sweden

Nils Svangård NILS.SVANGARD@GU.SE
Department of Applied Information Technology, University of Gothenburg, Sweden

David Lindström GUSDAVI04@STUDENT.GU.SE
Department of Applied Information Technology, University of Gothenburg, Sweden

Joscha Bach JOSCHA@BACH.AI
Evolutionary Dynamics, Harvard University, Cambridge, USA

Bas Steunebrink BAS@NNAISENSE.COM
NNAISENSE, Lugano, Switzerland

Editor: Pei Wang

Abstract

A computational model for artificial animals (animats) interacting with real or artificial ecosystems is presented. All animats use the same mechanisms for learning and decision-making. Each animat has its own set of needs and its own memory structure that undergoes continuous development and constitutes the basis for decision-making. The decision-making mechanism aims at keeping the needs of the animat as satisfied as possible for as long as possible. Reward and punishment are defined in terms of changes to the level of need satisfaction. The learning mechanisms are driven by prediction error relating to reward and punishment and are of two kinds: multi-objective local Q-learning and structural learning that alter the architecture of the memory structures by adding and removing nodes.

The animat model has the following key properties: (1) *autonomy*: it operates in a fully automatic fashion, without any need for interaction with human engineers. In particular, it does not depend on human engineers to provide goals, tasks, or seed knowledge. Still, it can operate either with or without human interaction; (2) *generality*: it uses the same learning and decision-making mechanisms in all environments, e.g. desert environments and forest environments and for all animats, e.g. frog animats and bee animats; and (3) *adequacy*: it is able to learn basic forms of animal skills such as eating, drinking, locomotion, and navigation.

Eight experiments are presented. The results obtained indicate that (i) dynamic memory structures are strictly more powerful than static; (ii) it is possible to use a fixed generic design to model basic cognitive processes of a wide range of animals and environments; and (iii) the animat framework enables a uniform and gradual approach to AGI, by successively taking on more challenging problems in the form of broader and more complex classes of environments.

Keywords: animats, homeostatic decision-making, structural learning, local Q-learning.

1. Introduction

The physicist P.J. van Heerden characterized *intelligent behavior* thus (Wilson, 1986):

> Intelligent behavior is to be repeatedly successful at satisfying one's psychological needs in diverse, observably different, situations on the basis of past experience [...]

This characterization makes no distinction between human and non-human animals. Moreover, it makes no distinction between body and mind, provided *psychological needs* is understood broadly enough: clearly, an animal's ability to satisfy its psychological needs depends on both its neural and non-neural anatomy. It is in line with that view on neuroscience that dismisses the distinction between body and mind to regard the nervous and endocrinic systems as physical control systems of the animal body (Bear, Connors, and Paradiso, 2015). Finally, this characterization makes no distinction between social and non-social intelligence (Insa-Cabrera, 2016). At an abstract level, living and dead objects play exactly the same role from an animal's perspective: both are parts of the external world, both generate sensory input, and both are objects that the animal interacts with and collects data about.

A strategy for understanding and simulating human intelligence is to follow the simple logic that humans are animals. According to that view psychology is a special case of ethology and human intelligence is a special case of animal intelligence. S.W. Wilson defines artificial animals, or *animats* thus, quoting from (Wilson, 1986):

1. The animal exists in a sea of sensory signals. At any moment, only some signals are significant; the rest are irrelevant.

2. The animal is capable of actions (e.g., movement) which change these signals.

3. Certain signals (e.g., those attendant on consumption of food) or their absence (e.g., those relating to pain) have special status.

4. He acts, both externally and internally, so as approximately to optimize the rate of occurrence of the special signals.

S.W. Wilson also suggested the *animat path to AI* with the goal of creating AI by modeling animal intelligence (Wilson, 1991). Animal intelligence has been studied and modeled extensively in comparative psychology and artificial-life (Langton, 1997; Tuci et al., 2016). To be of direct relevance to AGI, those models should (i) be computational, (ii) represent agents individually rather than collectively, and (iii) represent cognitive development explicitly. These criteria rule out the vast majority of models, including ecosystem models (Bolker, 2008) in which populations of animals are modeled by numbers, representing e.g. sex, age, and biomass distributions – whether those models are simulation-based (Christensen and Walters, 2004) or analytical (Caswell, 2001). Evolutionary game theory sometimes models agents at the individual level as e.g. finite or cellular automata; however, these models are of limited interest to AGI so long as they are static: i.e., no learning takes place at the individual level (Lindgren and Verendel, 2013).

Nervous systems are ubiquitous in the animal kingdom and play a fundamental role in animal intelligence. Successful artificial neural network techniques include deep learning (LeCun, Bengio, and Hinton, 2015; Schmidhuber, 2015), Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) with the ability to store arbitrary values for arbitrary

long time-periods, and Neural Turing machines, with their great generality, but also slow convergence and need for external memory resources (Zaremba and Sutskever, 2015).

Neuroplasticity refers to the capacity of animals to alter their nervous systems in response to changes in the environment. Not only does the connectivity between neurons change, but neurons may also be added and removed continuously in a life-long process (Draganski and May, 2008). Artificial neural network models are frequently based on static architectures that are only plastic in the sense that their connectivity patterns develop over time. Several neural network models also allow nodes to be added, however. For instance, the cascade-correlation architecture adds one hidden neuron at the time (Fahlman and Lebiere, 1990) and the progressive neural networks grow new columns while retaining previously acquired knowledge (Rusu et al., 2016). In the opposite direction there are regularization techniques (Goodfellow, Bengio, and Courville, 2016) and pruning methods (Wolfe et al., 2017) for reducing the size of neural networks, while improving generalization.

Reinforcement learning – adapting behavior in response to reward and punishment – is known to be a ubiquitous learning mechanism in the animal kingdom. Moreover, several neural mechanisms for learning through reward and punishment have been identified (Niv, 2009). In computer science, reinforcement learning algorithms are powerful tools for learning and decision-making in a general setting (Sutton and Barto, 1998). Q-learning is a basic algorithm for learning an optimal policy from experience in any environment that can be described by a Markov Decision Process (Watkins, 1989). Local Q-learning is a variation of Q-learning that is used to merge Q-values collected by multiple agents into a global Q-value (Russell and Zimdars, 2003). Multi-objective reinforcement learning can be used when the reward signal is a vector rather than a scalar (Roijers et al., 2013). A way of representing conjunctions of basic features in reinforcement learning is described in (Buro, 1998). A more sophisticated method along similar lines is the U-tree algorithm that builds decision-trees for state representations on the fly (Jonsson and Barto, 2001).

Homeostatic agents have homeostasis, or need satisfaction, as their sole objective, i.e. to regulate their homeostatic variables and thus survive as long as possible (Keramati and Gutkin, 2011; Yoshida, 2017). Homeostatic decision-making combines naturally with models for hormonal control (Avila-García and Cañamero, 2005), cognitive modulation (Bach, 2015), and personality traits (Bouneffouf, Rish, and Cecchi, 2017). Moreover, homeostatic agents can be naturally linked to reinforcement learning by defining reward as the difference in need status from one time to another. In the case of multiple homeostatic variables, or multiple needs, it may be natural to use multi-objective reinforcement learning (Roijers et al., 2013) rather than combining multiple reward signals into one with the help of a merge function. For an animal with water and energy as needs, no quantity of water can compensate for an energy deficit and *vice versa*. Running out of any of those resources may lead to instant death. Moreover, there is an upper bound to the amount of energy and water that a given animal can ingest.

There are several agent architectures that could potentially be used for modeling artificial animals, although they were not primarily constructed for this purpose, e.g. OpenCog (Goertzel, Pennachin, and Geisweiller, 2014), AERA (Nivel et al., 2013), MicroPsi (Bach, 2015), and NARS (Hammer, Lofthouse, and Wang, 2016).

Section 2 describes our strategy for AGI. Sections 3 defines our variety of graphs that are used for modeling memory structures. Section 4 defines the animats. Section 5 presents the

learning mechanisms and Section 6 the decision-making mechanisms of the animats. Section 7 presents the results of eight simulations of different animats in different environments. Section 8 contains a brief discussion about potential applications of the animat model along with some ethical considerations. Section 9, finally, draws some conclusions.

2. Strategy

Our main strategy towards AGI is an elaboration of the *animat path to AI*. Our approach is based on the idea that the nervous systems of animals ranging from fruit flies to elephants are essentially conformationd by the same fundamental mechanisms, many of which have already been identified and modeled computationally.

Our model makes use of ideas from several research fields. From *reinforcement learning*, we use elements of local Q-learning, multi-objective reinforcement learning, and the U-tree algorithm that were mentioned above. Our home-made single agent version of local Q-learning avoids the problem that AGI agents are unlikely to visit exactly the same state more than once (Wang and Hammer, 2015).

From *deep learning* we use the idea of hierarchical knowledge structures, although our memory structures have dynamic architectures and no weights on the connections. Our concept formation mechanisms have certain elements in common with the ontology building mechanisms outlined in (Adams and Burbeck, 2012). It also refines concepts gradually, similarly to version space models for concept learning (Mitchell, 1978), although our model differs in several ways, e.g. by supporting sequences. Our motivational system is a variation of the one used in MicroPsi2 (Bach, 2015). With inspiration from *radical constructivism* (Von Glasersfeld, 1995; Thórisson, 2012), we start from an arbitrary memory structure, e.g. encoding seed knowledge or just a *tabula rasa*, and gradually learn, by adding and removing nodes.

We use a memory model based on transparent neural networks (Strannegård and Nizamani, 2016) that we simply call *graphs*. The benefit of these graphs lies in their relatively transparent semantics, which makes it comparatively easy to define efficient rules for developing memory structures gradually. A fundamental learning mechanism is to add nodes to the memory whenever a sufficiently large prediction error occurs with respect to need satisfaction. The graph model enables our system to support several types of learning, including online, one-shot, and transfer learning. The decision-making algorithm has the permanent goal of keeping all the needs as satisfied as possible for as long as possible.

Our perspective on motivation and emotion largely follows the MicroPsi model (Bach, 2009) and its theoretical underpinnings, including Psi-theory (Dörner, 2001). Motivation is described as a set of physiological, social and cognitive needs, which the agent must keep in homeostasis. Different models can have different sets of needs. Each need has a target and current value; their difference manifests as an "urge" signal informing the agent's decision-making. All actions are directed either toward the satisfaction of a need (appetitive) or the avoidance of its frustration (aversive). Even serendipitous behavior serves needs: e.g., exploration (reduction of uncertainty) and competence (skill acquisition). The satisfaction of a need creates a pleasure signal proportional to the rate of satisfaction, amount, and relative importance of the need. The frustration of a need creates a pain signal. Pleasure and pain act as reinforcement signals for learning.

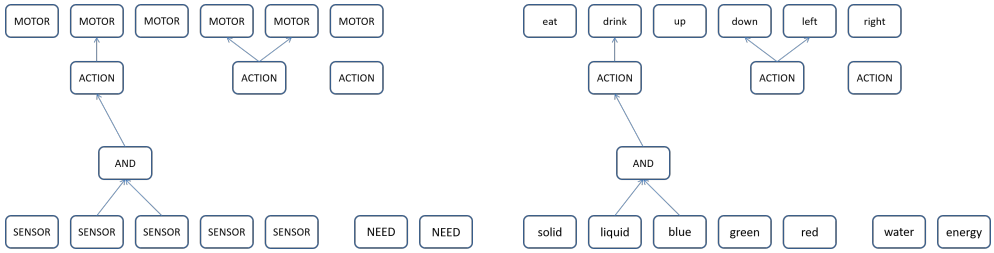

Figure 1: The left panel shows a graph. ACTION nodes may be connected to 0, 1, 2, or more MOTOR nodes. The ACTION node not connected to any MOTOR node represents passivity. The right panel shows the same graph with node names displayed. This graph has a reflex to drink whenever it encounters a blue liquid.

3. Graphs

In this section we introduce the graphs that are used for modeling memory structures.

Definition 1 (Graph) *A* graph *comprises:*

- *A set of nodes. Each node has exactly one type and a unique name. The types are NEED, SENSOR, AND, SEQ, MOTOR, and ACTION.*

- *A set of arrows (directed edges from nodes to nodes).*

NEED nodes are required to have fan-in 0 and fan-out 0. SENSOR-nodes have fan-in 0 and MOTOR nodes have fan-out 0. In other cases the fan-in and fan-out are unrestricted. An arrow pointing from an ACTION node or to a SEQ node may be labeled by a natural number.

The natural numbers are used for indicating temporal order, as will be clarified below. Figure 1 gives an example of a graph. Many more examples will be given below.

Intuitively, SENSOR nodes model sensors: receptor cells with ion channels sensitive to e.g. cold temperature, mechanical pressure, or acidity. NEED nodes model interoceptors: receptor cells that measure the level of satisfaction of various needs. For instance the NEED nodes may represent water (osmoceptors), energy (insulin receptors), protein (amino-acid receptors), oxygen (CO_2 receptors), integrity (nociceptors), sleep (melatonin receptors), heat (thermoreceptors), proximity (pheromone receptors), and affiliation (oxytocin receptors). MOTOR nodes model muscle-controlling motor neurons. AND nodes are used for modeling the usual big conjunction (that can have several conjuncts). SEQ nodes are used for modeling sequences. A SEQ node is activated if its incoming nodes have been activated in the order indicated by the annotation on the corresponding arrows. ACTION nodes model neurons that in turn activate motor neurons, either in parallel or in sequence. In parallel case, no annotations are used. In the sequences case, the natural number annotation is used to indicate order of execution of the MOTOR nodes. ACTION nodes can be activated either directly in the form of reflexes (arrows in the graph), or indirectly via decision-making, as we will see in Section 6. Arrows model synaptic connections that propagate activity through the graph.

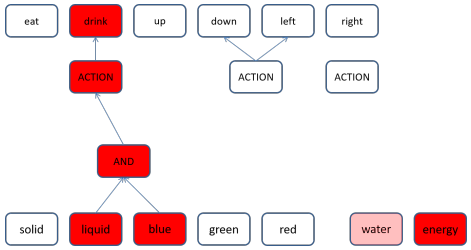

Figure 2: Activity propagation in the same graph as above. Bright red represents activity 1; white represents activity 0. Shades between white and red represent activity in $(0,1)$. In this graph, a drinking reflex is triggered by a blue liquid.

In the animat model, time is modeled in discrete time steps or *ticks*. At each tick, input activity is transmitted from the environment to the animat in the form of input activity to the SENSOR and NEED nodes of the animat's graph.

Definition 2 (Input) *An* input *to the graph G is an assignment of real values in $[0,1]$ to the NEED nodes and $\{0,1\}$ to the SENSOR nodes of G.*

Equivalently, the SENSOR nodes are assigned Boolean values. This input is the basis of the animat's perception. The input signals to the SENSOR nodes are propagated through the graph as follows:

- An AND node is true at t if all its conjuncts are true at t.
- A SEQ node b is true at t if the following holds for all the children b_i of b: the node b_i was true at time $t-k_i$, where k_i is the annotation on the arrow leading from b_i to b.
- A MOTOR node m is true at t if the following holds: (i) m is connected to an ACTION node a via an arrow without annotation such that a was activated at t, or (ii) m is connected to an ACTION node a via an arrow with annotation k such that a was activated at $t-k$.
- An ACTION node is true at t either if (i) it has an incoming arrow from a node that is true at t (reflex), or (ii) a is selected at t by the animat's policy, as specified in Section 6.

Figure 2 shows an example of activity propagation.

To illustrate the expressive power of our graph model, consider a piece of music M for a simple digital piano. Let there be one SENSOR node for each key on the piano; then M can be identified with a finite stream of inputs to the SENSOR nodes, while repeated AND and SEQ nodes represent chords and melodies respectively. In a similar way, for any given piece of music played on the same piano, there is an ACTION node that reproduces or plays this particular piece given that the MOTOR nodes represent the piano's keys.

Next we shall define the notion of top activity, which plays a key role in both decision-making and learning.

Definition 3 (Perception nodes) *A node labeled SENSOR, AND, or SEQ is called a* perception node*.*

Definition 4 (Top-active node) *Node b is top active at time t if:*

- *b is a perception node.*
- *b is active at t.*
- *No perception node b' that is also active at t is above b.*

We use the notation $TA(t)$ for the set of top active nodes at t.

Figure 2 provides an example, where the red AND node is the only top-active node. In general, many nodes can be top active at the same time. Intuitively, the top-active nodes describe how the present moment is perceived in terms of the perception nodes of the memory structures, at the maximum amount of detail.

4. Animats

In this section we will define the animats and the ecosystems where they live.

Definition 5 (Experience values) *There are two types of experience values:*

1. *A local Q-value is a real value $Q_i(b, a)$ that reflects the expected response to the status of need i when performing action a given that node b is top active.*
2. *A local R-value is a real value $R_i(b, a)$ that reflects the reliability of the value $Q_i(b, a)$.*

Definition 6 (Parameters) *A* parameter *is a real variable that regulates learning and decision-making. The parameters that we will use include learning rate (α), discount rate (γ), and exploration rate (ϵ).*

Definition 7 (Conformation) *A* conformation *is a subset of $\mathbb{R}^3$.*

conformations are used for specifying locations and positions of objects and animats of different kinds. In block worlds, for example, the conformations may be radically simplified. For instance, to specify the conformation of a rigid object or animat in a blocks world, it may suffice with two or three integer coordinates. Now we are ready to define the animats.

Definition 8 (Animat) *An* animat *consists of:*

- *a graph*
- *an activity pattern on this graph*
- *experience values*
- *parameters*
- *a conformation.*

Figure 3: The world consists of blocks of type "water", "grass", "sand", etc. The population consists of cattle and sheep animats.

Animats can model animals such as frogs and bees, or control artificial agents, such as robots. The activity pattern is included in this definition in order to enable perception of temporal sequences. We will soon give several examples of animats, but first let us define their environments.

Definition 9 (Object) *An* object *consists of:*

- *a type: a natural number. The types could, e.g. represent rock, earth, sand, air, or water.*

- *a conformation.*

Definition 10 (Ecosystem) *An* ecosystem *is a pair* $(\mathcal{A},\mathcal{O})$*, where* $\mathcal{A}$ *is a set of animats and* $\mathcal{O}$ *is a set of objects, such that all conformations are pairwise disjoint.*

Figure 3 shows an ecosystem. Several examples of ecosystems are given in Section 7.

Algorithm 1 shows how the animats are being updated at every tick. The algorithm is generic: it is the same for all animats. The conformation of the animat is not updated by this procedure. Instead it is updated by the physical laws of the environment, whether they are modeled as in the case of computer simulations or real as in the embodied case.

Algorithm 1: Main loop of the animat update algorithm.

```
Input: An initial animat A
while alive do
    Read the SENSOR values
    Update the perception node activity
    Make a decision
    Update the action activity
    Read the STATUS values
    Update the experience values
    Update the graph
end
```

In the next two sections we will describe the learning and decision-making steps of this algorithm in detail.

5. Learning

In this section we will describe the learning mechanisms of the animat.

5.1 Local Q-learning

Definition 11 (Status) *The* status *of the NEED node i of the animat A at time t, $x_{i,t}$, is defined as the input to i at t.*

An animat with NEED nodes "water" and "energy" could have $x_{water,t} = 0.8$ and $x_{energy,t} = 0.6$. The following measure reflects the overall well-being of an animat at a given moment.

Definition 12 (Vitality) *The* vitality *of the animat A at time t is defined as*

$$\min_{i \in NEED} x_{i,t}.$$

An animat with $x_{water,t} = 0.8$ and $x_{energy,t} = 0.6$ has vitality 0.6 at t. If the vitality reaches 0, we say that the animat *dies*. The learning and decision-making mechanisms of the generic animat were designed with long-term vitality as the one and only goal.

Definition 13 (Rewards) *The* reward *of the animat A at time $t+1$ with respect to the NEED node i is defined as $r_{i,t+1} = x_{i,t+1} - x_{i,t}$.*

Definition 14 (Reliability) *The* reliability *of the finite data set D is defined as $Rel(D) = 1/(SD(D)+1)$. Here SD is the standard deviation.*

We write a_t for the action that is performed at time t. Now we shall define the *local Q-values* $Q_{i,t}(b,a)$ and the *local reliability values* $R_{i,t}(b,a)$. Next we shall define the local Q-values and R-values. The definition of the R-values is a first attempt to capture the intuition that the more fluctuating a certain local Q-value has been, the less it can be trusted. Experience will tell whether this definition would benefit from being modified.

Definition 15 (Q-values and R-values) *At $t = 0$ we proceed as follows. Let*

$$Q_{i,0}(b,a) = 0 \text{ and } R_{i,0}(b,a) = 1$$

for all perception nodes b, ACTION nodes a, and NEED nodes i.

At $t+1$ we proceed as follows. If $b \notin TA(t)$ or $a \neq a_t$, then we let $Q_{i,t+1}(b,a) = Q_{i,t}(b,a)$. If $b \in TA(t)$, then we let

$$Q_{i,t+1}(b,a_t) = Q_{i,t}(b,a_t) + \alpha \cdot (r_{i,t+1} + \gamma \cdot \Delta),$$

where Δ is

$$\max_{a \in Actions} \left[\frac{\sum_{b' \in TA(t+1)} Q_{i,t}(b',a) \cdot R_{i,t}(b',a)}{\sum_{b' \in TA(t+1)} R_{i,t}(b',a)} \right] - Q_{i,t}(b,a_t).$$

Here α and γ are parameters for learning rate and discount rate, respectively. Also let $R_{i,t+1}(b,a)$ be

$$Rel(\{Q_{i,t'}(b,a) : t' \leq t+1, a = a_{t'} \text{ and } b \in TA(t')\}).$$

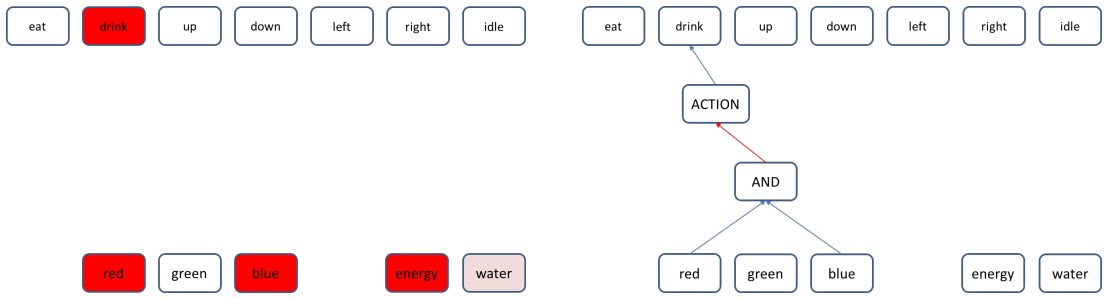

Figure 4: Left: an empty graph with only SENSOR, NEED, and MOTOR nodes. Right: the same graph with a belief added encoding the experience of drinking when the sensors "blue" and "red" are active.

5.2 Structural learning

Imagine an animat whose nervous system is as shown to the left in Figure 4. Suppose further that the animat perceives "red" and "blue" and attempts the action "drink". It is surprised as its energy level rises. The surprise should trigger learning; but what should be learned? One option is to add the belief:

Action "drink" in state "blue AND red" leads to reward. (1)

The result is shown to the right in Figure 4. Another option would be to be more specific and add the belief:

Action "drink" in state "blue AND red AND NOT green" leads to reward. (2)

The better choice is option (1): it is more general and, therefore, potentially more useful – albeit at the risk of overgeneralizing . It *could* be the case that the action "drink" in the state "blue AND red AND green" leads to punishment instead. Mechanisms are required that enable one to form more fine-grained beliefs as the need arises: i.e., to handle exceptions to general principles. Now let us make the intuition about surprise more precise.

Definition 16 (Surprise) *The* surprise *of a perception node b at time $t+1$ w.r.t. the NEED node i is defined as follows:*

$$z_{i,t+1}(b) = |Q_{i,t+1}(b,a_t) - Q_{i,t}(b,a_t)|$$

Definition 17 (Surprised) *An animat is* surprised *at time $t+1$ if $z_{i,t+1}(b) > Z$, for some NEED node i and perception node b such that $R_{i,t}(b,a) > R$. Here Z and R are parameters regulating concept formation.*

When the animat is surprised, a new node will be added to the graph. The surprise indicates that the animat needs a more fine-grained ontology to be able to identify similar situations in the future.

Definition 18 (Node candidate) *A* node candidate *is an expression of the form*

- *b AND b′, where b,b′ ∈ G are perception nodes and b AND b′ ∉ G, or*

- *b SEQ b′, where b,b′ ∈ G are perception nodes and b SEQ b′ ∉ G.*

The node candidates do not belong to the graph, but they have local Q-values and R-values that are initiated and updated just like the local values of the perception nodes of the graph.

Suppose the animat gets surprised at $t+1$. Then the learning algorithm will consider the possibility of adding a new node. Let i be a randomly selected NEED node subject to surprise at $t+1$.

First, the algorithm explores the benefit of adding an AND node. To that end it searches for a node candidate b AND b' such that (i) both b and b' were top-active at t, and (ii) the prediction error

$$|Q_{i,t+1}(b\ AND\ b', a_t) - Q_{i,t}(b\ AND\ b', a_t)|$$

is minimal. If this prediction error is sufficiently small, the node b AND b' is added to the graph.

Second, if no AND node is added, the algorithm proceeds by exploring the benefit of adding a SEQ node. To that end it searches for a node candidate b SEQ b' such that (i) b was top-active at $t-1$, (ii) b was top-active at t, and (iii) the prediction error

$$|Q_{i,t+1}(b\ SEQ\ b', a_t) - Q_{i,t}(b\ SEQ\ b', a_t)|$$

is minimal. If this prediction error is sufficiently small, the node b SEQ b' is added to the graph. Whenever a new node is added to the graph, new node candidates are formed (by Definition 18).

Now, let us turn to the forgetting rule, which works in the opposite direction and removes nodes.

Definition 19 (Forgetting rule) *Suppose $Q_i(b\ AND\ b', a) \approx Q_i(b, a)$, for all actions a and NEED nodes i. Then delete the node $b\ AND\ b'$ from the graph.*

Let us illustrate the point of structural learning and top activity with a little story. Imagine a bear that tasted berries for the first time: berries that happen to be blueberries. Then the bear experienced reward via a rise in its blood-sugar level. The consequent positive surprise caused it to form the node "berry", e.g. as a small sphere, and set $Q_{energy}(berry, eat)$ to a positive number. Thus the following belief was added to its memory:

Action "eat" when perceiving "berry" leads to reward. (3)

The belief (3) guided the bear to many good meals of blueberries. One day the bear encountered red berries for the first time. Guided by its belief (3) it ate those berries. But the red berries happened to be poisonous red honeysuckle berries, that caused the bear to vomit and thus lose energy. In response to this negative surprise it added the following belief to its memory:

Action "eat" when perceiving "berry AND red" leads to punishment. (4)

Over the next few months, the bear encountered red berries of different types many times and avoided eating them because of its belief (4). When winter approached, the bear was very hungry. It came to a forest with red berries that happened to be shiny red cranberries. Since it was so hungry it decides to take a risk despite belief (4) and try those berries. Then it got a big positive surprise, since the berries were delicious. This surprise caused it to add the following belief to its memory:

Action "eat" when perceiving "berry AND red AND shiny" leads to reward. (5)

From that day forth, no surprise caused it to modify its beliefs about berries anymore. At that point it had learned everything it needed to know about which berries are edible. In fact, the beliefs (3)-(5) capture all it needs to know about berries to live a long life in that forest. This story illustrates the point that memory formation should be exactly as fine-grained as required from the perspective of need satisfaction. It also illustrates the point of basing decisions on the most detailed memories that are activated in a given situation, in other words the top-active nodes.

6. Decision-making

Definition 20 (Policy) *Let λ be a real-valued parameter for present-future value trade-off and let*

$$\pi(t) = \underset{a \in Actions}{\operatorname{argmax}} \left[\min_{i \in NEED} (x_{i,t} + \lambda \cdot \frac{\sum_{b \in TA(t)} Q_{i,t}(b,a) \cdot R_{i,t}(b,a)}{\sum_{b \in TA(t)} R_{i,t}(b,a))}) \right].$$

The policy selects actions aimed at keeping the vitality of the animat as high as possible, for as long as possible. It weighs up the animat's present status with expected status changes in the future. These expectations are in turn weighted by their estimated reliability. An animat with the two needs "energy" and "water" will be likelier to drink if water is its most urgent need. On the other hand, if its experience indicates that it would lose large quantities of energy by doing so, it might nevertheless refrain.

The decision-making algorithm is ε-greedy, where $\varepsilon \in [0,1]$. With probability ε it explores by activating a random set of MOTOR nodes (with higher probability for smaller sets) and with probability $1-\varepsilon$ it exploits by following the policy $\pi(t)$.

7. Results

We have implemented the prototype system *Generic Animat*, which is available at `https://github.com/ni1s/animats`. The system is a simplification of the model described in the previous section and it is integrated with Minecraft via the Malmo interface (Johnson et al., 2016). At present, the implementation does not support multiple animats.

To define an animat, an initial graph must be specified. Any graph will do. For instance, it can be a *tabula rasa* with NEED, SENSOR and MOTOR nodes only. It can also be a graph with complex memory structures and reflexes. Once it has been initialized, the animat can then be put into an arbitrary world, where it will learn and make decisions continuously and automatically. To measure the performance of an animat in a given world, we study how its vitality develops over time. In this section we shall give several examples illustrating how

the *Generic Animat* can learn to eat, drink, move, navigate, and conceptualize its world by forming spatial and temporal patterns.

7.1 Sheep world 1

Here we show that a sheep animat can learn to eat and drink. Consider a sheep with two needs: energy and water. Its world is shown in Figure 5 and its initial memory in Figure 6. The interaction dynamics between animat and world are described in tables 1, 2, and 3.

Figure 5: A world containing three blocks: grass, sand, and water.

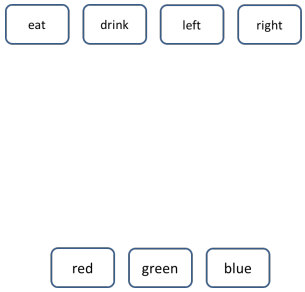

Figure 6: The memory of the sheep animat at the start. It has two NEED nodes: "energy" and "water"; three SENSOR nodes: "red", "blue" and "green"; and four ACTION nodes.

	Grass	Sand	Water
red	0	1	0
green	1	0	0
blue	0	0	1

Table 1: Perception of the sheep animat.

energy	Grass	Sand	Water
eat	0.1	-0.05	-0.05
drink	-0.05	-0.05	-0.001
left	-0.001	-0.001	-0.001
right	-0.001	-0.001	-0.001

Table 2: Status changes for the NEED node "energy".

We ran a simulation with the sheep animat. Figure 7 shows the memory after convergence and Figure 8 shows its vitality curve.

water	Grass	Sand	Water
eat	-0.001	-0.05	-0.05
drink	-0.05	-0.05	0.1
left	-0.001	-0.001	-0.001
right	-0.001	-0.001	-0.001

Table 3: Status changes for the NEED node "water".

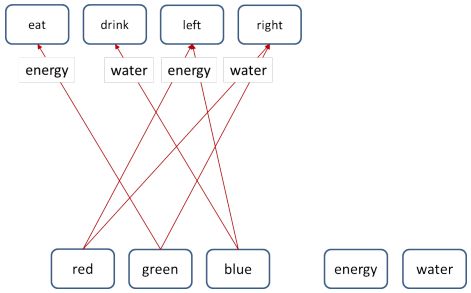

Figure 7: The memory of the sheep animat after convergence at time 20. The red arrows indicate preferred actions in response to each need.

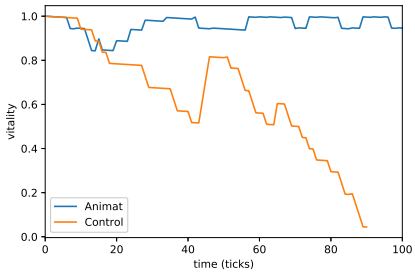

Figure 8: Vitality curve of the sheep, where vitality is the minimum of "energy" and "water". "Control" shows the same animat performing random actions. Multi-objective reinforcement learning helps the animat to learn a strategy that lets it survive: alternating between eating grass and drinking water depending on the most pressing need. "Control" quickly declines and dies.

7.2 Sheep world 2

Here we illustrate the benefit of structural learning in the case of AND node addition. Consider a sheep animat that lives in the world shown in Figure 9. Figures 6 and 10 show its memory at start and after convergence, respectively. Figure 11 shows how its vitality develops over time.

7.3 Sheep world 3

Here we illustrate the benefit of structural learning in the case of SEQ node addition. Again we consider a sheep that drinks and grazes, but this time the sheep lives in a world that

Figure 9: The green block represents grass that is good to eat and the blue block water that is good to drink. The middle block represents a swamp where eating or drinking leads to vomiting and thus to decreased water and energy levels.

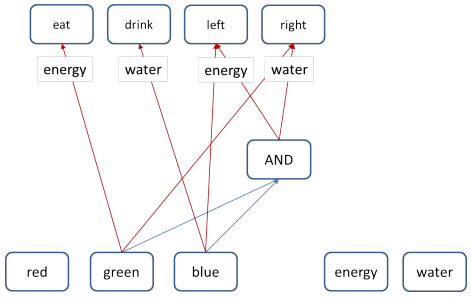

Figure 10: The memory after convergence (at time 25). The labels on the red arrows indicate the preferred actions for the different needs when the lower node is top active. The AND node that was added automatically enables the animat to survive.

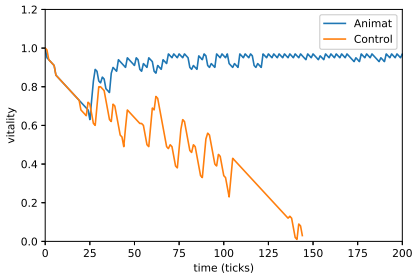

Figure 11: "Animat" is the sheep animat. "Control" is similar, but its dynamic concept formation is switched off. They both start out with a blank slate. "Animat" adds an AND node at time 25. It manages to survive, while "Control" dies.

contains both good water and bad, poisonous water. The problem is that the sheep cannot differentiate directly between the good and the bad water with its sensors. By learning that the bad water always appears in a certain context, in this case close to sand, the animat can learn to avoid drinking it.

The world is shown in Figure 12. The animat starts with the same memory as in the previous example (Figure 10). It adds the node "red SEQ blue" the first time a red block is encountered (one-shot learning). Figure 13 shows its vitality curve.

7.4 Plankton world

Here we model a copepod (a planktonic crustacean) that lives in the ocean and feeds on planktonic algae; its world is shown in Figure 14 and its memory development in Figure

Figure 12: This world is a long path that begins with Water and Grass blocks, where the animat can learn to eat and drink. Then come the Poison blocks for the first time. Each Poison block has a Sand block to its left. This enables animats that are capable of sequence learning to differentiate between Water and Poison.

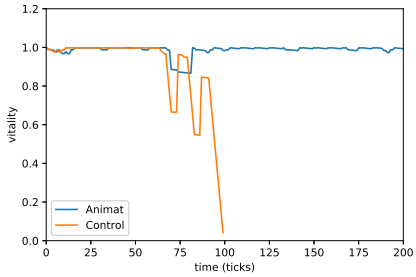

Figure 13: "Animat" is the sheep animat. "Control" is similar, but it has its capacity to add SEQ nodes switched off. The animats start with a blank slate. "Animat" adds a SEQ node at time 75. It survives, while "Control", unable to learn sequences and contexts, dies at time 100.

15. The interactive dynamics between animat and world is described in Table 4. Figure 16 shows how the animat learns to survive.

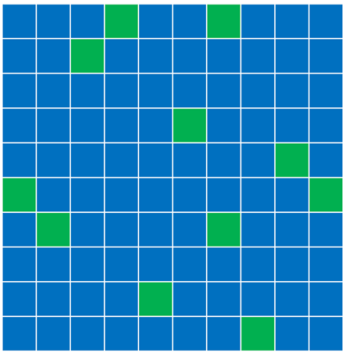

Figure 14: The copepod's initial world, including two types of blocks: water (blue) and algae (green). The world has a torus topology obtained by "gluing" together first the upper edge with the lower edge and then the left edge with the right edge.

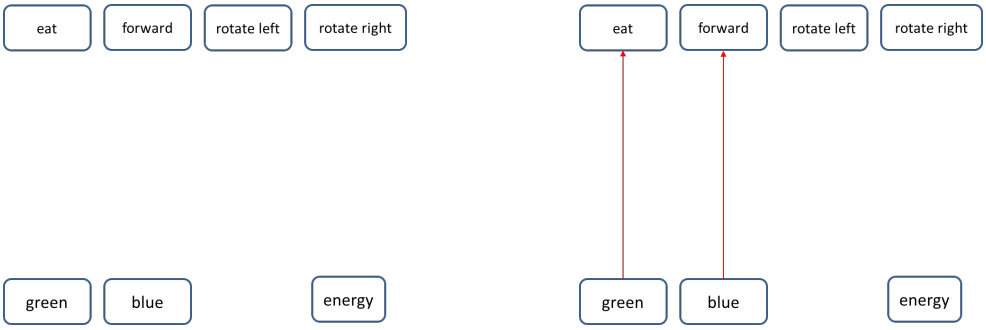

Figure 15: The copepod animat's memory development. Its initial memory – a *tabula rasa* – is at left. The animat has one NEED node: "energy"; and two SENSOR nodes: "blue" and "green". It has four ACTION nodes. In this case, the MOTOR nodes can be identified with the ACTION nodes. At right is the memory after about twenty time steps. The red arrows indicate preferred actions.

	water	algae
blue	1	0
green	0	1

energy	water	algae
eat	-0.01	0.3
forward	-0.001	-0.001
rotate left	-0.001	-0.001
rotate right	-0.001	-0.001

Table 4: Interaction dynamics for the copepod animat. At left is the perception matrix: how input to the SENSOR nodes is generated. At right is the reward matrix: how "energy" values (which can never exceed 1) change depending on the situation. The negative numbers represent metabolism costs. The effects of actions are as expected: e.g., the action "forward" causes the animat to move forward, the action "eat" causes an algae block to be replaced by a water block.

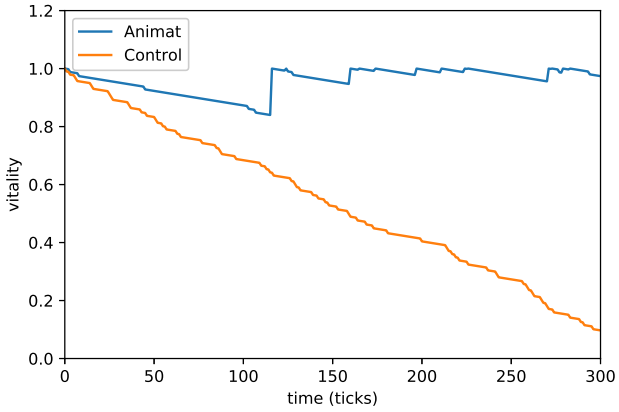

Figure 16: Vitality curve for the copepod animat ("vitality" here is the same as "energy level"). "Control" shows the same animat performing random actions. Reinforcement learning helps the animat gradually to learn a strategy that lets it survive: eating when it encounters green blocks and preferring forward motion to rotation. Note how quickly "control" declines and dies.

7.5 Frog world

Here we show a frog animat that learns how to move. First, let us consider a frog that lives in the world shown in Figure 17. Figure 18 shows the initial memory of the frog.

Figure 17: The frog world. A one-dimensional world of green blocks.

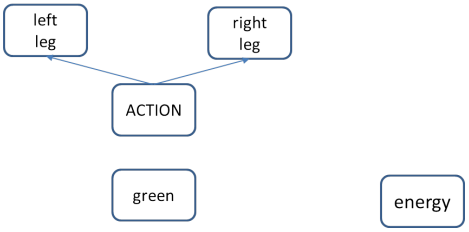

Figure 18: The initial memory of the frog. The frog has three actions: it can extend its left hind leg only, its right hind leg only, or both its hind legs (jump).

The frog consumes energy from metabolism at each time tick. It can only gain energy by moving to new blocks and ingesting the food that is available there. It can only move to a new block by jumping, i.e. by extending both hind legs simultaneously. The result of a 100-tick simulation is shown in Figure 19 and Figure 20.

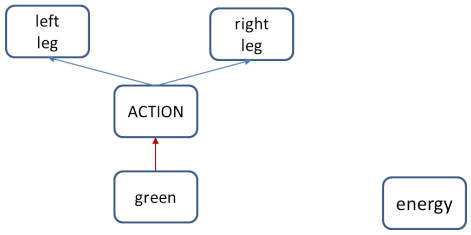

Figure 19: The memory of the frog after convergence. Convergence happens at time 3 when the frog has learned to prefer jumping.

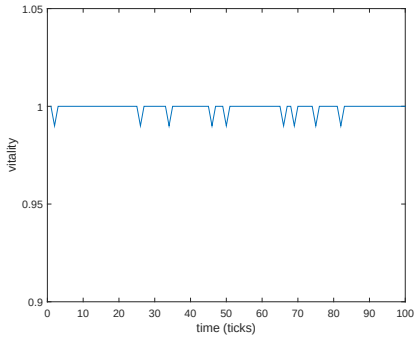

Figure 20: Vitality curve of the frog. The frog has learned how to jump after three ticks. The sporadic dips in the vitality curve are due to exploration of non-optimal actions.

7.6 Toad world

Here we consider a toad that lives in the same world as the frog. The toad cannot jump; it can only move forward by alternately extending its hind legs. If a single leg is extended then the other leg is automatically folded. An action to extend an already extended leg results in that leg continuing to be extended. The toad is equipped with proprioception sensors that indicate which hind legs are folded. The initial memory of the toad is shown in Figure 21.

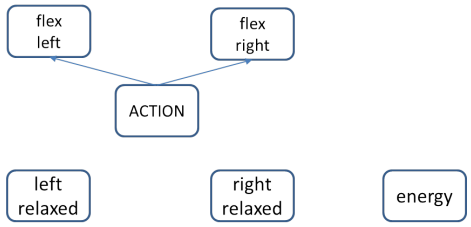

Figure 21: The initial memory of the toad. The toad has two sensors for proprioception and the same actions as the frog.

The result of a 100-tick simulation is shown in Figure 22 and Figure 23.

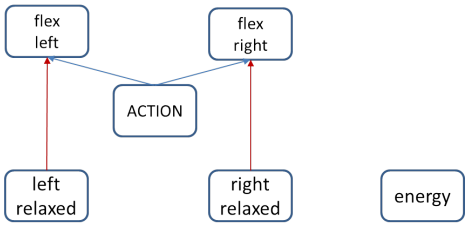

Figure 22: The memory of the toad after convergence. It has learned how to crawl.

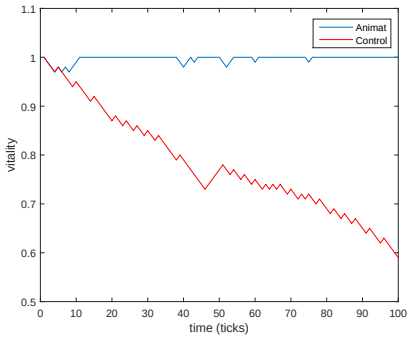

Figure 23: Vitality curve of the toad. "Animat" is the toad with proprioception sensors and "Control" is similar but with no proprioception sensors.

7.7 Fish world

Here we consider a fish animat that learns to swim toward reward gradients and away from punishment gradients. We use two similar worlds to show this. The worlds are built from seven blocks which give energy to or take energy from the animat. The animat has one SENSOR node for blue, one NEED node for energy, and MOTOR nodes for moving left and right. The animat starts on the leftmost block and it bounces back when it hits a wall. Both worlds are as shown in Figure 24. The rewards are shown in Table 5. Thus, in World 1 the animat starts in the "good" end and in World 2 the animat starts in the "bad" end of the world. The results of two 500-tick simulations are shown in Figure 25 for World 1 and Figure 26 for World 2.

Figure 24: The fish world.

	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
World 1	0.001	0	-0.001	-0.002	-0.003	-0.004	-0.005
World 2	-0.005	-0.004	-0.003	-0.002	-0.001	0	0.001

Table 5: Energy changes when the fish reaches a given block by moving left or right.

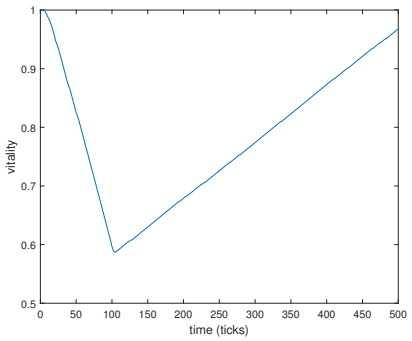

Figure 25: Vitality curve for World 1. The fish starts to the left at the "good" end and learns to stay there.

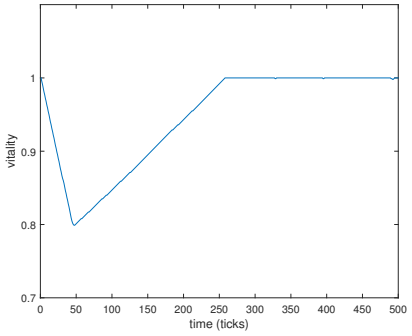

Figure 26: Vitality curve for World 2. The fish starts to the left at the "bad" end and learns to move to the "good" end.

7.8 Bee world

Here we show that the generic animat can learn to navigate like a Braitenberg vehicle. We model a bee that navigates in a landscape with scenting flowers.

Consider a bee animat living in the world shown in Figure 27. When the bee visits a flower it collects the nectar, transforming the flower into grass. Each flower diffuses a scent into its surroundings. The intensity of the scent from a flower at a given distance follows the inverse-square law ($intensity \propto 1/distance^2$). The attractive and repulsive flowers have, respectively, positive and negative scents. The initial memory of the bee is shown in Figure 28. The energy level of the bee changes depending on three factors: metabolism, scent intensity and whether nectar is collected.

We ran a 200-tick simulation of the bee in the bee world. Figure 29 shows a mid-simulation plot of how the bee has moved in the world so far. The result of the simulation is shown in Figure 30 and Figure 31.

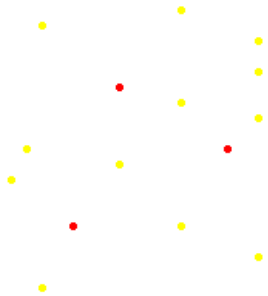

Figure 27: The bee world. This world is a two-dimensional 20x20 array of blocks with a torus topology. There are three types of blocks: attractive flowers, repulsive flowers, and grass. There are 12 attractive flowers (yellow dots) and 3 repulsive flowers (red dots).

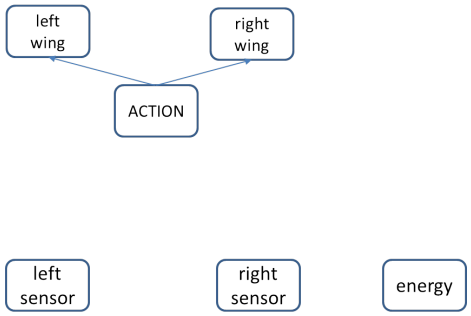

Figure 28: The initial memory of the bee. The bee has two sensors sensitive to the direction of the scent gradient of nectar from flowers in its 9x9 blocks neighborhood. One is active if the scent gradient is to the left of the animat or within 22.5 degrees to the front-right, and the other is active if the scent gradient is to the right of the animat or within 22.5 degrees to the front-left. Thus, there is an overlap of the sensitivity of the "left" and "right" sensors and they will both be active if the scent vector is within +/- 22.5 degrees of the forward direction of the animat.

8. Discussion

8.1 Training animats

Most animals are autonomous in the sense that they are able to live and reproduce without any interaction with humans. Moreover, they have no interfaces that enable human engineers to reprogram them or human users to provide them with direct commands in a formal language. Yet many animals can be trained by humans for various purposes. For instance, dogs can be trained to become service dogs, guard dogs, avalanche dogs, narcotics dogs, sled dogs, or hunting dogs. In that case the training targets the animals' existing reward systems and their ability to learn by association and positive reinforcement (Rooney and Cowan, 2011). Dogs can also live in the wild similarly to wolves and survive without any interaction whatsoever with humans.

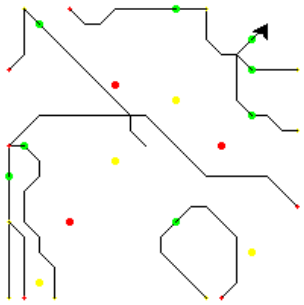

Figure 29: A mid-simulation snapshot of the bee's trajectory. The arrowhead represents the current position and heading of the bee. The black line shows the trajectory of the bee from the start. The yellow dots represent attractive flowers. Green dots represent flowers from which the bee has collected nectar. Red dots represent repulsive flowers.

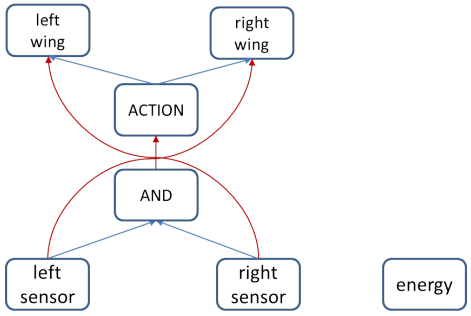

Figure 30: The memory of the bee after convergence. The bee has learned to turn left when the "left" sensor is active and right when the "right" sensor is active. An AND node was added and the bee has learned to move forward when this node is active.

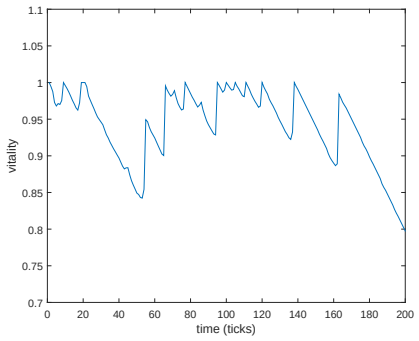

Figure 31: Vitality curve of the bee. An AND node is added to the graph at time 7. The first five times nectar is collected are at times 7, 17, 53, 64 and 75. The last available nectar is collected at time 161.

Just like animals, animats can develop with any amount of interaction with humans. Moreover, they can develop in a computer simulated environment or have a robotic body and

develop in the real world. For instance, they can first be trained in a simulated environment and then inserted into a robotic body and operate in the real world.

Now let us consider for a while how animats can be used for serving humans. To be able to train the animat, the human trainer must have access to its reward system. One way is to give reward and punishment to one of the existing homeostatic variables, e.g. energy. Another way is to add a homeostatic variable that is directly controlled by humans. If desirable, this homeostatic variable can be given priority over the other homeostatic variables by re-scaling the values of the other homeostatic variables. Reward and punishment that target this additional homeostatic variable might then be provided by the human users, e.g. in the form of verbal praise and blame. Then the animat might be trained similarly to dogs.

8.2 Ethical aspects

The alignment problem formulated by Bostrom (Bostrom, 2014) concerns alignment of the behavior of artificial agents with human desiderata, e.g. expressed as utility or reward functions. This problem is intrinsically hard in the real world because of the unpredictability of what situations might arise in the future and the difficulty of making changes to the utility function once an autonomous agent is operational (Taylor et al., 2016). Computer simulation combined with reward and punishment given by humans might contribute strongly to alignment, but just like in the case of dogs there is no guarantee for reliability. Generally speaking, autonomous agents are safety-critical, whether they operate in a software environment or in a physical body. Thus one would ideally want to prove that these agents always (or never) do certain actions in certain situations. In fact, testing is not enough, since it is impossible in practice to test all the relevant cases. It has been argued that neural networks have a low degree of transparency and are unsuitable for safety-critical decision-making (Nusser, 2009). There seem to be no efficient methods for formalizing and proving safety properties of neural networks at present. This suggests that neural networks alone are not suitable at this stage for controlling autonomous agents. Logic-based systems, on the other hand, have a long tradition of formal verification and safety-critical decision-making. They do not seem to match the neural networks when it comes to learning, however. The animat model uses logic-based concepts and so they have a relatively transparent semantics that lends itself to theorem proving. Thus one may, e.g. impose the restriction on the animat that a proof must have been found before the animat is allowed to change its graph architecture.

9. Conclusion

We have described a computational model for artificial animals by defining generic rules for perception, learning, and decision-making. These rules constitute a basic developmental model for arbitrary animats in arbitrary ecosystems. We presented several experiments with a prototype implementation of the animat model. The experiments showed how animats can adapt to different ecosystems and learn to survive by, e.g. starting from an empty memory structure and gradually populating it with nodes and connections. The results obtained indicate the generality of the model and show that the animats are capable of learning to eat, move, and navigate in a number of different environments. It was also shown that

the ability to form new nodes sometimes means the difference between life and death. The results obtained suggest that dynamic architectures are more powerful than static architectures. The animat model described was intended as a proof of concept. Clearly, it can be developed further in several directions. For instance, the structural learning rules could be improved and straightforward probabilistic rules that form memories at random moments could be added. Second, to enable symbolic learning and reasoning, a working memory and a rewrite engine could be added, possibly along the lines of (Strannegård et al., 2016). Third, the exploration techniques could be refined, e.g. by avoiding actions that are believed to bring strong punishment while exploring. Finally, the animat model needs to be tested in much more complex and challenging ecosystems. In particular, it must be tested in ecosystems with multiple animats.

To conclude, the animat path to AI seems very promising and natural to us and we believe in the strategy of gradually refining a generic animat model so that it can survive in ever larger and more complex classes of ecosystems. By restricting attention to a general type of problem that faces all animals – that of surviving (and reproducing) in different ecosystems – one may approach the AGI challenge in a uniform and stepwise fashion. This is the type of problems that all animals were evolved to solve. The animat path to AI is in sharp contrast to strategies for AGI that focus on problem domains such as IQ-tests, theorem proving, language comprehension, and strategic board games.

Acknowledgement

This research was supported by the Torsten Söderberg Foundation Ö110/17. C. S. is grateful to Martin Nowak for enabling a research visit to the Evolutionary Dynamics program at Harvard University.

References

Adams, S. S., and Burbeck, S. 2012. Beyond the Octopus: From General Intelligence toward a Human-like Mind. In *Theoretical Foundations of Artificial General Intelligence*. Springer. 49–65.

Avila-García, O., and Cañamero, L. 2005. Hormonal modulation of perception in motivation-based action selection architectures. In *Procs of the Symposium on Agents that Want and Like*. SSAISB.

Bach, J. 2009. *Principles of synthetic intelligence*. Oxford University Press.

Bach, J. 2015. Modeling motivation in MicroPsi 2. In *AGI 2015 Conference Proceedings*, 3–13. Springer.

Bear, M. F.; Connors, B. W.; and Paradiso, M. A. 2015. *Neuroscience*. Wolters Kluwer.

Bolker, B. M. 2008. *Ecological models and data in R*. Princeton University Press.

Bostrom, N. 2014. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.

Bouneffouf, D.; Rish, I.; and Cecchi, G. A. 2017. Bandit Models of Human Behavior: Reward Processing in Mental Disorders. In *AGI 2017 Conference Proceedings*, 237–248. Springer.

Buro, M. 1998. From simple features to sophisticated evaluation functions. In *International Conference on Computers and Games*, 126–145. Springer.

Caswell, H. 2001. *Matrix population models*. Wiley Online Library. Avaliable at https://www.sinauer.com/media.

Christensen, V., and Walters, C. J. 2004. Ecopath with Ecosim: methods, capabilities and limitations. *Ecological modelling* 172(2-4):109–139.

Dörner, D. 2001. *Bauplan für eine Seele*. Rororo. Rowohlt-Taschenbuch-Verlag.

Draganski, B., and May, A. 2008. Training-induced structural changes in the adult human brain. *Behavioural brain research* 192(1):137–142.

Fahlman, S. E., and Lebiere, C. 1990. The cascade-correlation learning architecture. In *Advances in neural information processing systems*, 524–532.

Goertzel, B.; Pennachin, C.; and Geisweiller, N. 2014. The OpenCog Framework. In *Engineering General Intelligence, Part 2*. Springer. 3–29.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hammer, P.; Lofthouse, T.; and Wang, P. 2016. The OpenNARS implementation of the non-axiomatic reasoning system. In *AGI 2016 Conference Proceedings*. Springer. 160–170.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Insa-Cabrera, J. 2016. *Towards a Universal Test of Social Intelligence*. Ph.D. Dissertation, Universitat Politècnica de València, Valencia, Spain.

Johnson, M.; Hofmann, K.; Hutton, T.; and Bignell, D. 2016. The Malmo platform for artificial intelligence experimentation. In *International joint conference on artificial intelligence (IJCAI)*, 4246.

Jonsson, A., and Barto, A. G. 2001. Automated state abstraction for options using the U-tree algorithm. In *Advances in neural information processing systems*, 1054–1060.

Keramati, M., and Gutkin, B. S. 2011. A reinforcement learning theory for homeostatic regulation. In *Advances in neural information processing systems*, 82–90.

Langton, C. G. 1997. *Artificial life: An overview*. MIT Press.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553):436.

Lindgren, K., and Verendel, V. 2013. Evolutionary Exploration of the Finitely Repeated Prisoners' Dilemma—The Effect of Out-of-Equilibrium Play. *Games* 4(1):1–20.

Mitchell, T. M. 1978. Version spaces: an approach to concept learning. Technical report, STANFORD UNIV, CALIF, DEPT OF COMPUTER SCIENCE.

Niv, Y. 2009. Reinforcement learning in the brain. *Journal of Mathematical Psychology* 53(3):139–154.

Nivel, E.; Thórisson, K. R.; Steunebrink, B. R.; Dindo, H.; Pezzulo, G.; Rodriguez, M.; Hernandez, C.; Ognibene, D.; Schmidhuber, J.; Sanz, R.; et al. 2013. Bounded recursive self-improvement. *arXiv preprint arXiv:1312.6764*.

Nusser, S. 2009. Robust Learning in Safety-Related Domains. *Machine Learning Methods for Solving Safety-Related Application Problems, Otto-von-Guericke-Universität Magdeburg*.

Roijers, D. M.; Vamplew, P.; Whiteson, S.; Dazeley, R.; et al. 2013. A Survey of Multi-Objective Sequential Decision-Making. *J. Artif. Intell. Res.(JAIR)* 48:67–113.

Rooney, N. J., and Cowan, S. 2011. Training methods and owner–dog interactions: Links with dog behaviour and learning ability. *Applied Animal Behaviour Science* 132(3):169–177.

Russell, S. J., and Zimdars, A. 2003. Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 656–663.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Schmidhuber, J. 2015. Deep Learning in Neural Networks: An Overview. *Neural Networks* 61:85–117.

Strannegård, C., and Nizamani, A. R. 2016. Integrating Symbolic and Sub-symbolic Reasoning. In *AGI 2016 Conference Proceedings*, 171–180. Springer.

Strannegård, C.; Nizamani, A. R.; Juel, J.; and Persson, U. 2016. Learning and Reasoning in Unknown Domains. *Journal of Artificial General Intelligence* 7(1):104–127.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT press.

Taylor, J.; Yudkowsky, E.; LaVictoire, P.; and Critch, A. 2016. Alignment for advanced machine learning systems. *Machine Intelligence Research Institute*.

Thórisson, K. R. 2012. A new constructivist AI: from manual methods to self-constructive systems. In *Theoretical Foundations of Artificial General Intelligence*. Springer. 145–171.

Tuci, E.; Giagkos, A.; Wilson, M.; and Hallam, J., eds. 2016. *From Animals to Animats. 1st International Conference on the Simulation of Adaptive Behavior*. Springer.

Von Glasersfeld, E. 1995. *Radical Constructivism: A Way of Knowing and Learning. Studies in Mathematics Education Series: 6.* ERIC.

Wang, P., and Hammer, P. 2015. Assumptions of Decision-Making Models in AGI. In *AGI 2015 Conference Proceedings*. Springer. 197–207.

Watkins, C. J. C. H. 1989. *Learning from delayed rewards*. Ph.D. Dissertation, King's College, Cambridge.

Wilson, S. W. 1986. Knowledge growth in an artificial animal. In *Adaptive and Learning Systems*. Springer. 255–264.

Wilson, S. W. 1991. The animat path to AI. In Meyer, J. A., and Wilson, S. W., eds., *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*.

Wolfe, N.; Sharma, A.; Drude, L.; and Raj, B. 2017. The Incredible Shrinking Neural Network: New Perspectives on Learning Representations Through The Lens of Pruning. *arXiv preprint arXiv:1701.04465*.

Yoshida, N. 2017. Homeostatic Agent for General Environment. *Journal of Artificial General Intelligence* 8(1).

Zaremba, W., and Sutskever, I. 2015. Reinforcement learning neural turing machines-revised. *arXiv preprint arXiv:1505.00521*.