Generic Animats

Claes Strannegård[1,2(✉)], Nils Svangård[3], Joscha Bach[4], and Bas Steunebrink[5]

[1] Department of Computer Science and Engineering,
Chalmers University of Technology, Gothenburg, Sweden
claes.strannegard@chalmers.se
[2] Department of Philosophy, Linguistics and Theory of Science,
University of Gothenburg, Gothenburg, Sweden
[3] Department of Applied Information Technology,
University of Gothenburg, Gothenburg, Sweden
nils.svangard@gu.se
[4] Evolutionary Dynamics, Harvard University, Cambridge, USA
joscha@bach.ai
[5] NNAISENSE, Lugano, Switzerland
bas@nnaisense.com

Abstract. We present a computational model for artificial animals (animats) living in block worlds, e.g. in Minecraft. Each animat has its individual sets of needs, sensors, and motors. It also has a memory structure that undergoes continuous development and constitutes the basis for decision-making. The mechanisms for learning and decision-making are generic in the sense that they are the same for all animats. The goal of the decision-making is always the same: to keep the needs as satisfied as possible for as long as possible. All learning is driven by surprise relating to need satisfaction. The learning mechanisms are of two kinds: (i) structural learning that adds nodes and connections to the memory structure; (ii) a local version of multi-objective Q-learning. The animats are autonomous and capable of adaptation to arbitrary block worlds without any need for seed knowledge.

Keywords: Autonomous agent · Dynamic graph · Multi-objective reinforcement learning · Structural learning · Need satisfaction

According to the South African physicist Pieter Jacobus van Heerden [20]:

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

Interpreted broadly, this characterization takes physiological, social, and cognitive needs into account - along with the body, since the body plays a central role in satisfying one's needs. It also applies to all animal species, not just humans. Moreover, it does not rely on human judgement as in the Turing test; or on human artifacts, as in standard IQ tests.

T. Everitt et al. (Eds.): AGI 2017, LNAI 10414, pp. 23–32, 2017.
DOI: 10.1007/978-3-319-63703-7_3

In artificial intelligence, deep Q-learning has seen great success in recent years [9,14]. One of the most prominent examples in the direction of general intelligence is the generic Atari-game player that learned to play 31 Atari games at super-human level [10]. For a discussion of some theoretical problems associated with deep Q-learning, see [19].

Graph structures that develop gradually have been used in finite automaton learning [2], cascade correlation networks [6], and deep network cascades [1].

This paper is about artificial animals (animats). These models have mainly been studied in the field of artificial life [18]. Section 1 describes our strategy for general intelligence. Section 2 describes our computational model. Section 3 presents the prototype implementation *Generic Animat* along with two examples illustrating the advantage of structural learning. Section 4 draws some conclusions.

The proposed computational model is partly a continuation of our previous work [4,12,16]. The mechanisms for local Q-learning and structural learning are novel as far as we know.

1 Strategy

Our approach to general intelligence is based on the idea that radically different nervous systems can be formed by the same underlying biological mechanisms, starting with different bodies and experiencing different sensory data. We model the following generic mechanisms for learning and decision-making, which are ubiquitous in the animal kingdom:

1. Decision-making that aims for the satisfaction of multiple physiological needs [13].
2. Reinforcement learning that strengthens/weakens behavior associated with reward/punishment [11].
3. Hebbian learning, captured in the popular phrase "cells that fire together, wire together" [3].
4. Sequence learning, which is Hebbian learning with signal delay taken into account [5].

In our model an animat may be defined by specifying its sets of needs, sensors, and motors. The animat then develops automatically by means of computational versions of the above-mentioned generic mechanisms for learning and decision-making.

To model the environments of the animats, we use the Minecraft computer game environment [8], putting the animats into the bodies of Minecraft animals such as sheep, rabbits, and wolves. Then we can study the animats as they strive to satisfy their needs, e.g. for company, grass, and drinking water.

2 Computational Model

This section presents the components of the computational model.

2.1 Worlds

Definition 1. *A* world *is a set of blocks. A* block *consists of:*

- *A block type (a natural number).*
- *A block position (a point in three-dimensional space* $\mathbb{Z}^3$*).*

2.2 Dynamic Graphs

To model memory structures of animats, we use labeled graphs extended with support for multi-objective reinforcement learning. The nodes of the graphs can be identified with formulas of temporal logic [7]. In particular we use the binary modal operator SEQ that enables the construction of sequences. The formula p SEQ q is true at time t if p is true at $t-1$ and q is true at t.

Definition 2. *A* dynamic graph *consists of:*

- *A set of* nodes *labeled SENSOR, STATUS, MOTOR, AND, OR, NOT, SEQ, or ACTION and optionally given a name.*
- *A set of* arrows*, i.e. a binary relation on the nodes. Arrows pointing to ACTION-nodes are labeled with local Q-values and R-values, as will be explained in Subsect. 2.6.*

Figure 1 shows a dynamic graph.

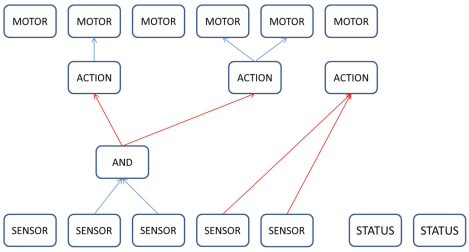

Fig. 1. An example of a dynamic graph with some annotation on the arrows omitted. Note that ACTION nodes may be connected to 0, 1, 2, or more MOTOR nodes. The ACTION node that is not connected to any MOTOR node represents passivity.

2.3 Activity

Definition 3. *An* activity *of dynamic graph* G *is an assignment of values in* $[0,1]$ *to the nodes of* G*, subject to the restriction that non-STATUS nodes must be assigned values in* $\{0,1\}$.

Figure 2 shows an activity. Time is modeled in discrete time steps or *ticks*. Input activity is transmitted from the environment to the SENSOR and STATUS nodes. Activity propagates to the other nodes as expected, except in the case of the ACTION nodes. The activity of ACTION nodes is determined by the policy given in Definition 12.

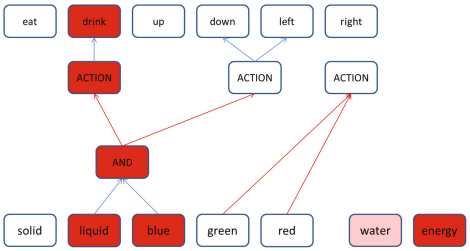

Fig. 2. An example of an activity pattern on a graph. This is the same graph as in Fig. 1, but with the optional node names displayed. Color represents activity 1; no color represents activity 0. The color shades on the STATUS nodes reflect their input values. (Color figure online)

2.4 Animats

Definition 4. *An* animat *consists of:*

- *A dynamic graph G.*
- *An activity of G.*
- *A position: a point in the space $\mathbb{Z}^3$.*

2.5 Top Activity

Definition 5 (Perception nodes). *A node labeled SENSOR, AND, OR, NOT, or SEQ is called a* perception node*.*

The following notion plays a key role in both decision-making and learning:

Definition 6 (Top-active node). *Node $b \in G$ is top active at time t if:*

- *b is a perception node.*
- *b is active at t.*
- *There is no blue arrow that starts in b and ends in some other perception node b' that is also active at t.*

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

Figure 2 offers 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 together constitute a description of the present situation in terms of the given memory structure, at the maximum possible level of detail.

2.6 Local Q-learning

We use a local variant of Q-learning for our generic learning and decision-making mechanism [17]. Since the animat has multiple needs to take into account in the general case, we work within the Multi-Objective Reinforcement Learning framework [13].

Definition 7 (Status). *The* status *of the STATUS node i of the animat A at time t, $x_{i,t}$, is defined as the input to i at t.*

An animat with STATUS 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 8 (Vitalit). *The* vitality *of the animat A at time t is defined as*

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

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

Definition 10 (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* $R_{i,t}(b,a)$.

Definition 11 (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 STATUS 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 $Q_{i,t+1}(b,a_t) = Q_{i,t}(b,a_t) + \alpha \cdot \Delta$, where

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

Here α and γ are parameters for learning rate and discount rate, respectively. Also let

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

Definition 12 (Policy). *Fix a real number λ and let*

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

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 $\pi(t)$.

2.7 Structural Learning

Definition 13 (Surprise). *The* surprise *of a perception node b at time $t+1$ w.r.t. the STATUS 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 14 (Surprised). *An animat is* surprised *at time $t+1$ if $z_{i,t+1}(b) > Z$, for some STATUS 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 15 (Node candidate). *A* node candidate *is an expression of the form*

- *b AND b', where $b, b' \in G$ are perception nodes and b AND $b' \notin G$, or*
- *b SEQ b', where $b, b' \in G$ are perception nodes and b SEQ $b' \notin G$.*

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 STATUS 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 15).

3 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 precious section and it is integrated with Minecraft via the Malmo interface [8]. To measure the performance of an animat in a given world, we study how its vitality develops over time. Cf. the quote by van Heerden at the beginning of the paper.

We shall give two minimalist examples showing that the ability to form new concepts can make the difference between life and death. In both examples the *Generic Animat* controls a sheep animat that needs to drink and graze to survive.

3.1 Learning Spatial Patterns

In this example we show the usefulness of adding AND nodes. The sheep animat lives in the world shown in Fig. 3. Figure 4 shows its memory development and Fig. 5 shows how its vitality develops over time.

Fig. 3. The Grass block is green and represents grass that is good to eat and the Water block is blue and represents water that is good to drink. The Swamp block is turquoise (blue and green) and represents a swamp where eating or drinking leads to vomiting and thus to a decrease in the water and energy levels. (Color figure online)

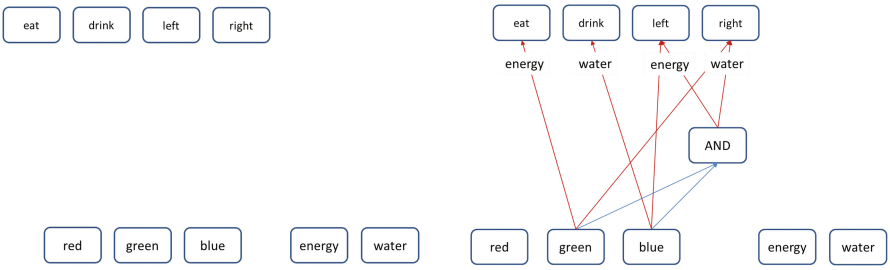

Fig. 4. The left panel shows the initial *blank slate* memory of the animat with two STATUS nodes: energy and water; three SENSOR nodes for colors; and four ACTION nodes. The right panel shows the memory after convergence (at time 25). The labels on the red arrows indicate the preferred actions for the different needs when the lower node is top active. The AND node that was added automatically enables the animat to survive.

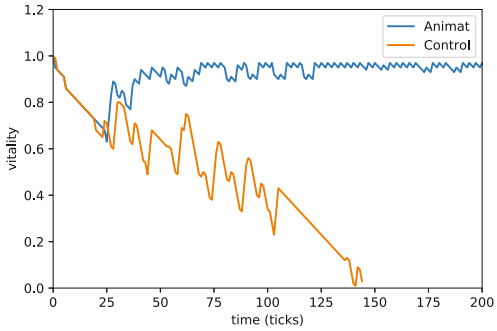

Fig. 5. Animat is the sheep animat. Control is similar, but its dynamic concept formation is switched off. They both start out with a blank slate. Animat adds an AND node at time step 25. It manages to survive, while Control dies.

3.2 Learning Temporal Patterns

In this example we show the usefulness of adding SEQ nodes. Again we consider a sheep animat that drinks and grazes, but this time the animat lives in a world that contains both good water and bad, poisonous water. The problem is that the animat cannot differentiate directly between the good and the bad water

Fig. 6. 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. 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.

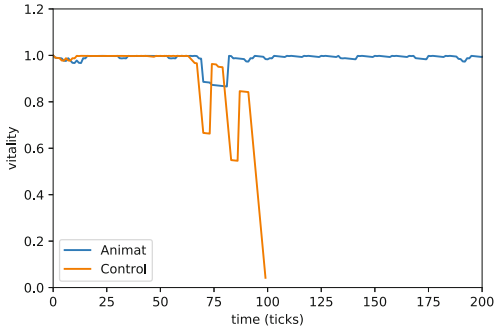

Fig. 7. Animat is the sheep animat. Control is similar, but it has its capacity to add SEQ nodes switched off. The animats start with a blank slate. Animat adds a SEQ node at time step 75. It survives, while Control, unable to learn sequences and contexts, dies at time step 100.

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 Fig. 6. The animat starts with the same memory as in the previous example (Fig. 4). It adds the node "red SEQ blue" the first time a red block is encountered. Figure 7 shows how its vitality develops over time.

4 Conclusion

We have presented a model of a generic animat that is based on several techniques, including dynamic graphs for memory representation; top activity for perception; reliability and Q-values for decision-making; and local Q-learning and structural learning for memory development.

The model was constructed with the goal of preserving full generality while keeping the computational complexity at a minimum level. To that end we avoided explicit representations of subsets of sensors. Instead we use top-active nodes that represent partially defined states. We designed the model so that it only adds nodes reluctantly in case it gets surprised with respect to reward or punishment. An additional way of reducing the computational complexity might be to use forgetting, as was done in [15].

Our generic model is capable of starting with an arbitrary dynamic graph – e.g. a blank slate – using structural learning and local Q-learning to build a memory structure that helps the animat keep its needs satisfied and survive. Our examples here were designed to illustrate how structural learning can make the difference between life and death. Our model is fully autonomous and fully versatile; it does not depend on "seed" knowledge of any kind. In that sense it possess a basic form of general intelligence.

Acknowledgement. This research was supported by The Swedish Research Council, grants 2012-1000 and 2013-4873. C.S. is grateful to Martin Nowak for enabling a research visit to Harvard University.

References

1. Angelova, A., Krizhevsky, A., Vanhoucke, V., Ogale, A.S., Ferguson, D.: Real-time pedestrian detection with deep network cascades. In: BMVC, pp. 32.1–32.12 (2015)
2. Angluin, D.: Finding patterns common to a set of strings. J. Comput. Syst. Sci. **21**(1), 46–62 (1980)
3. Baars, B., Gage, N.: Cognition, Brain, and Consciousness: Introduction to Cognitive Neuroscience. Academic Press, San Diego (2010)
4. Bach, J.: MicroPsi 2: the next generation of the MicroPsi framework. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 11–20. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35506-6_2

5. Bear, M.F., Connors, B.W., Paradiso, M.A.: Neuroscience. Wolters Kluwer, Hagerstown (2015)
6. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture (1990)
7. Gabbay, D.M., Hodkinson, I., Reynolds, M.: Temporal Logic Mathematical Foundations and Computational Aspects. Oxford University Press, Oxford (1994)
8. Johnson, M., Hofmann, K., Hutton, T., Bignell, D.: The malmo platform for artificial intelligence experimentation. In: International Joint Conference on Artificial Intelligence (IJCAI), p. 4246 (2016)
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
10. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
11. Niv, Y.: Reinforcement learning in the brain. J. Math. Psychol. **53**(3), 139–154 (2009)
12. Nivel, E., Thórisson, K.R., Steunebrink, B.R., Dindo, H., Pezzulo, G., Rodriguez, M., Hernandez, C., Ognibene, D., Schmidhuber, J., Sanz, R., et al.: Bounded recursive self-improvement. arXiv preprint arXiv:1312.6764 (2013)
13. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R., et al.: A survey of multi-objective sequential decision-making. J. Artif. Intell. Res. (JAIR) **48**, 67–113 (2013)
14. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
15. Strannegård, C., Cirillo, S., Wessberg, J.: Emotional concept development. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) International Conference on Artificial General Intelligence, pp. 362–372. Springer, Switzerland (2015)
16. Strannegård, C., Nizamani, A.R.: Integrating symbolic and sub-symbolic reasoning. In: Steunebrink, B., Wang, P., Goertzel, B. (eds.) AGI -2016. LNCS, vol. 9782, pp. 171–180. Springer, Cham (2016). doi:10.1007/978-3-319-41649-6_17
17. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (1998)
18. Tuci, E., Giagkos, A., Wilson, M., Hallam, J. (eds.): From Animals to Animats, 1st International Conference on the Simulation of Adaptive Behavior. Springer, Switzerland (2016)
19. Wang, P., Li, X.: Different conceptions of learning: function approximation vs. self-organization. In: Steunebrink, B., Wang, P., Goertzel, B. (eds.) AGI -2016. LNCS, vol. 9782, pp. 140–149. Springer, Cham (2016). doi:10.1007/978-3-319-41649-6_14
20. Wilson, S.W.: Knowledge growth in an artificial animal. In: Narendra, K.S. (ed.) Adap. Learn. Syst., pp. 255–264. Springer, New York (1986)