DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch061

Evolution of Social Representation in Neural Networks

Solvi Arnold, Reiji Suzuki and Takaya Arita

Graduate School for Information Science, Nagoya University, Japan
solvi@alife.cs.is.nagoya-u.ac.jp

Abstract

This paper describes an Artificial Life approach to Theory of Mind (ToM), the ability to employ mental representations of other minds in order to understand or anticipate the behaviour of others. We designed a model in which a population of neural network (NN) agents evolve the ability to predict, on basis of observation of past behaviour, others' future behaviour in novel circumstances. As agent behaviour is guided by private mental states, invisible to the predicting agent, this task forces agents to go beyond imitation and repetition of fit responses, requiring them to gain some degree of *insight* into the partner agent's internal configuration by observation of their externally visible behaviour. As such, this learning ability cannot be captured with conventional learning algorithms based on rewards or examples. We find that NNs equipped with neuromodulation mechanisms can be evolved to perform favourably on this task. The resulting networks are seen to behave as though they have a primitive form of first order ToM.

Introduction

Theory of Mind (ToM) is the ability to employ mental models (representations) of other minds, in order to understand or anticipate the behaviour of others (Premack and Woodruff, 1978). The adaptive advantages of ToM are likely to be a driving factor in the evolution of cognition that recognizes others as well as itself as intentional agents. While ToM has become a hot topic in cognitive psychology and related fields, the phenomenon of "mirror neurons" (neurons that activate both when a given action is performed and when the same action is observed) has become a hot topic in cognitive neuroscience. Many researchers are intuitively inclined to link these two phenomena, thinking of mirror neurons as a neural basis for ToM, but the relation between the two remains murky. As such it seems potentially informative to try and evolve ToM-like abilities in neural systems.

Representation is a tough issue in connectionist AI, but mental representation of other minds presents a special challenge in at least two aspects: (1) Other minds are themselves capable of representing, leading to recursive and reflexive scenarios such as mind X representing a mind Y that itself represents mind X (see also Dennett, 1987). This point in particular complicates the connection with mirror neurons, which so far have not been observed to engage in recursive mirroring (although some have theorized about recursive functionality, see Gallese, 2007). (2) Other minds are invisible: we cannot see the minds of others, we can only guess at the existence of other minds via observation of behaviour. This point has direct implications for learning about other minds: One might learn about another's behaviour via direct observation of that behaviour, but for learning about another's mind one needs forms of learning ability that incorporate inference from externally visible behaviour to (invisible) mental states. This sort of learning is difficult to capture with traditional AI conceptualizations of learning. Indeed, while computational work on ToM exists, the mechanisms for representing other minds are usually explicitly given and fixed (see e.g. Takano and Arita, 2006; Noble et al., 2010) (placing the focus on recursion depth instead). In this research we instead aim to let such mechanisms evolve from scratch, using a minimalistic evolutionary neural network model.

Model

Agents are implemented as neural networks (NNs). Network architecture is evolved using a basic Genetic Algorithm. Agents interact in pairs. During its lifetime, each agent is part of multiple pairings. In each pair, there is a fixed role division: one agent acts at zero-order ToM (L_0), meaning it ignores the other agent and simply reacts on basis of the state of the environment and its own mental state, and the other agent acts at first-order ToM (L_1), meaning it tries to anticipate the behaviour of the L_0 agent (at present, the L_1 agent is simply tasked with predicting the L_0 agent's behaviour). Each pair interacts for a set number of time-steps.

Our model is not intended to capture any specific social interaction scenario in particular. Instead we take a more abstract approach, in which the logic that determines the fitness payoff for performing a given action in a given state is generated randomly for each experiment (i.e. the fitness function is randomly generated for each run of the model). The idea is that if arbitrary fitness functions can be handled successfully, then the model has generality. Thus there is no concrete "task" to solve, there are merely *environmental states*, *mental states*, *actions*, and a randomly generated *base logic* that relates these elements.

Environmental state: bit-string of length N_e (set to 3 in the experiments discussed in this paper). The environmental state is shared between interacting agents (i.e. both agents see the same state). The environmental state changes every time-step. Each pair of agents sees each environmental state exactly once, in random order.

Mental state: bit-string of length N_m (set to 3 in the experiments discussed in this paper). Each agent has a private mental state, invisible to its interaction partner. Mental states remain constant over the course of the interaction of an agent pair.

Action: bit-string of length N_a (set to 3 in the experiments discussed in this paper). At each time-step, each agent outputs an action.

Base logic: generates the optimal L_0 action choice for each (environmental state, mental state) pair. The base logic abstractly represents social scenarios. The base logic is implemented as a neural network, identical in kind to the NNs used for the agents, although with a few restrictions (we detail the way the base logic is generated after we explain the agent NN architecture below).

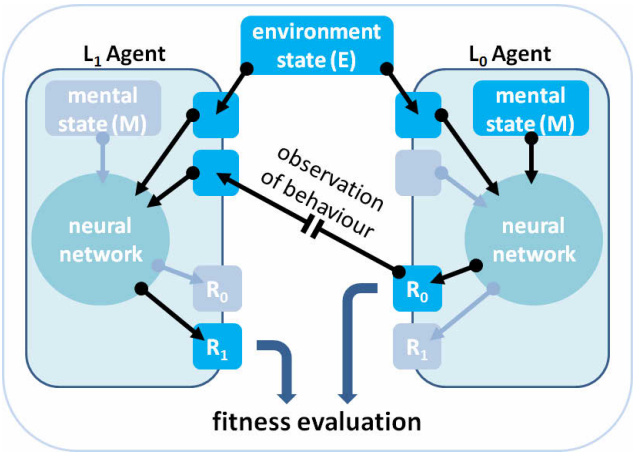

Figure 1. Schematic of agent interaction. The L_0 agent computes its action (R_0) from the (shared) environmental state and its (private) mental state. The L_1 agent computes a prediction (R_1) of this action. After the prediction is made, R_0 is revealed to the L_1 agent as data to drive its learning process.

Fitness scores for the action choices of the agent performing the L_0 role are calculated as proximity to the optimal action as generated by the base logic, normalized to the [-1, +1] interval. Meanwhile, fitness scores for the action choices of the agent performing the L_1 role are calculated as proximity to the action choice of the L_0 agent. As such, the L_1 agent must try to predict the action of the L_0 agent, but the action choice of the L_0 agent depends on the L_0 agent's mental state, which is invisible to the L_1 agent. Herein lies the challenge: in order for the L_1 agent to be able to predict the L_0 agent's future moves under future environmental states, the L_1 agent must infer the L_0 agent's mental state from the L_0 agent's action choices under the current and preceding environmental states. By observing both the L_0 agent's action choice and the environmental state that led the L_0 agent to choose that action, the L_1 agent has the necessary information to infer the L_0 agent's mental state, and on the basis thereof it can predict the L_0 agent's behaviour under other environmental states.

Network species

We use a slightly unusual type of NN, which gives a central position to propagation order. A network consists of a list of neurons and a set of connections. Propagation simply follows the list order. The genome encodes for each connection the list-indices of the pre-synaptic neuron and the post-synaptic neuron. If the index of the post-synaptic neuron is smaller (or equal) to the index of the pre-synaptic neuron, then the connection runs against the propagation direction and is thus treated as a recurrent connection (meaning activation sent over it arrives at the next time-step). Otherwise, it runs along the propagation direction and is treated as a regular connection (meaning activation sent over it arrives at the same time-step). This approach avoids the trouble of deriving propagation order in free-form evolvable NN architectures, and facilitates later implementation of endogenously controlled propagation loops for recursive ToM.

The neuron list is composed of seven sections: $2 \cdot N_m$ neurons for mental state input, $2 \cdot N_e$ neurons for environmental state input, $2 \cdot N_a$ neurons for partner action input, H hidden neurons, N_a neurons for L_0 action output, again H hidden neurons, and finally N_a neurons for L_1 action output. Connections between two neurons inside one and the same input or output section are not allowed. We provide two neurons for each input bit. Activation of one neuron signals a 1 value and activation of the other neuron signals a 0 value for the bit. These values can have very different implications, so we input them separately. When a NN acts as L_0, its responses are read from the first set of output neurons, and when it acts as L_1, the second set is read. When reading out responses at the output neurons, we translate neural activation values into binary values by converting negative values into zeroes and positive values into ones. Figure 2 shows the basic architecture (for $N_m = N_e = N_a = 1$ and $H = 2$). In the experiments discussed in this paper we used the following settings: $N_m = N_e = N_a = 3$ and $H = 16$, making for a total of 56 neurons. Given that connectivity is evolved, it is very well possible for neurons to not be included in the circuitry, meaning that the "effective" net size will generally be smaller than 56. The maximum number of connections is limited to 100.

A neuron's activation is computed from the activation of the neurons that project to it, using a slight modification of the standard hyperbolic tangent activation function.

Activation function:

$$A_j = N_j \cdot \tanh\left[0.5 \cdot \sum\left(W_{ij}^{A} \cdot A_i\right)\right] + b_j$$

Where A_i is activation at neuron i, W_{ij}^{A} is the weight of an activatory connection from i to j, b_j is the (genetically encoded) activation bias for neuron j, and N_j is the neurotransmitter value at neuron j. N_j defaults to 1, but can be lowered if the neuron has any incoming *neurotransmitter connections*. Activation received over such connections is added to the N_j value. N_j is clipped to the [0, 1] range before the activation function is applied, and reset to 1 after propagation. This neurotransmitter logic is included to provide a simple mechanism for blocking signal transmission, which can simplify some neural computations (e.g. xor without hidden neurons). While theoretically speaking this does not expand the functionality

of the network species, we do find inclusion of such a neurotransmitter system to improve evolvability.

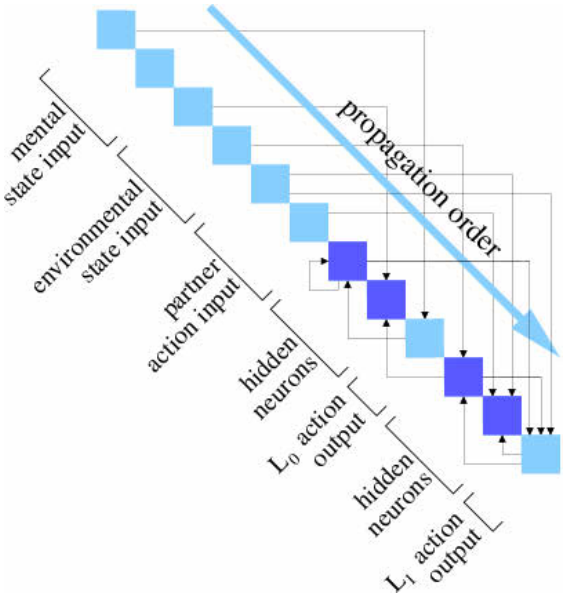

Figure 2. Neural network architecture concept. Networks consist of seven sections, with the size of each section determined by model parameters (see text). Propagation order is fixed. Connections running against the propagation order are treated as recurrent connections.

Evolution of learning ability is made possible using neuromodulation (Soltoggio et al., 2008). Similar techniques have previously been employed to evolve spatial representation ability in NNs (Arnold et al., 2012, 2013), so it stands to reason that it might allow for evolution of social representation ability as well. The basic idea is to introduce a special connection type that lets neurons send modulatory signals to one another, and to let these signals control connection weight change. This allows for evolution to shape the weight update dynamics of the networks by shaping the modulatory connectivity, which provides a basis for endogenously controlled behaviour change, i.e. a basis for learning ability. So in addition to their activation value, neurons have a modulation value, and in addition to standard activatory connections, there are modulatory connections. If neuron i has a modulatory connection to neuron j, then activation at i leads to modulation at j. Neurons' modulation values are computed in similar fashion to their activation values, but without involvement of neurotransmitter values.

Neuromodulation function:

$$M_j = \tanh\left[0.5 \cdot \sum\left(W_{ij}^{M} \cdot A_i\right)\right]$$

Where M_i is modulation at neuron i, and W_{ij}^{M} is the weight of a modulatory connection from i to j.

Weight updates are computed from the activation and modulation values at the pre- and post-synaptic neurons as follows:

$$\Delta W_{ij} = A_i^{gij0} \cdot A_j^{gij1} \cdot M_i^{gij2} \cdot M_j^{gij3}$$

Where ΔW_{ij} is the change in the weight of the connection from neuron i to neuron j, and $g_{ij}0\ldots3$ are binary genes that encode inclusion/exclusion of each term in the update function for this specific connection. Connection weights are clipped to [-1, +1].

Weight updates are only performed when the network plays the role of an L_1 agent (the behaviour target for L_0 agents is static, so no learning is required there). When an agent plays the L_1 role, propagation is performed twice. The first time, only the environmental state is given on the input neurons, propagation is performed, and the action prediction is read out on the L_1 output neurons. Then the environmental state and the actual action choice of the L_0 agent are given as input, propagation is performed, and connection weights are updated. In the second propagation round, output is ignored.

Genetic Algorithm

Networks are evolved using a Genetic Algorithm, with mutation but no crossover. After each generation, agents are sorted by fitness, after which the worst performing two thirds of the population is replaced with copies of the best performing one third, to which then mutation is applied. Mutations can alter the following properties:

Connections
- Pre- and post-synaptic neuron indices
- Type (activatory, neurotransmitter, modulatory)
- Update rule genes ($g_{ij}0\ldots3$)
- Existence

Neurons
- Activation bias value

The "existence" property is used for addition and removal of connections (technically speaking there are always 100 connections in the network, but those with the existence property set to false are skipped over when the propagation logic is performed).

We additionally include some special mutation operators for modifying neural pathways, such as “inserting” a neuron into a pathway (given a connection from neuron x to neuron z, this operator picks a random neuron y and then replaces the original xz connection with a xy and a yz connection), and “removing” a neuron from a pathway (given a neuron y, it finds a connection projecting from some x to y and a connection from y to some z, and replaces the xy and yz connections with a single xz connection). Necessity of such operators has not been investigated here.

When mutating a network, we first pick a mutation rate using the following rule:

$$\text{rate} = 16 \cdot R(0,1)^6$$
$$\text{rate}_{\text{neuron}} = \text{rate} \,/\, \text{neurons}$$
$$\text{rate}_{\text{connection}} = \text{rate} \,/\, \text{connections}$$

Where R(0,1) generates random numbers in the interval [0,1], *neurons* is the number of neurons in the agent networks (56 in our experiments), and *connections* is the maximum number of connections in the agent networks (100 in our experiments). The resulting $\text{rate}_{\text{neuron}}$ and $\text{rate}_{\text{connection}}$ values are then used as the mutation probabilities per neuron and per connection, respectively. This slightly convoluted system for computing mutation rates is intended to help evolution by producing a good mixture of heavily and mildly mutated individuals, and was found to work better than fixed mutation rates.

Generating the base logic

As noted above, we use random base logics instead of specific social scenarios. Base logics are implemented as special instances of the neural network architecture described above, and generated using the same genetic algorithm, just using a different fitness function than used when evolving agent NNs. The base logic serves as the target phenotype for L_0 agents (i.e. fitness of L_0 behaviour is judged as proximity to the base logic). As the base logic is just a special exemplar of the same neural network species, it takes environmental and mental states as input and returns actions as output. The fitness function used for generating base logics assesses the suitability of this input-output mapping as a social task, using the product of two values:

Environmental state relevance: for each possible mental state, we count the number of different action outputs that can be obtained by varying the environmental state. This gives an indication of the relevance of the environmental state for optimal action choice. If different environmental states do not lead to different optimal actions, then little can be learned by observing another agent in different environmental states. Thus the base logic should have a high value for environmental state relevance, to make learning *possible*.

Mental state relevance: for each possible environmental state, we count the number of different action outputs that can be obtained by varying the mental state. This gives an indication of the relevance of the mental state for optimal action choice. If different mental states do not lead to different optimal actions, then there is no need for L_1 agents to learn about the mental state of the L_0 agent. Thus the base logic should have a high value for mental state relevance as well, to make learning *necessary*.

Expressed in formulaic form:

$$\text{fitness}_{\text{logic}} = \sum_{\text{mst}}^{\text{m}} \text{actions(m)} \cdot \sum_{\text{env}}^{\text{e}} \text{actions(e)}$$

Where mst is the set of possible mental states, env is the set of environmental states, actions(m) is the number of distinct actions that can be obtained for mental state m by varying the environmental state, and actions(e) is the number of distinct actions that can be obtained for environmental state e by varying the mental state.

While we use the same network architecture, some limitations are imposed when a network is used as base logic. First off, the base logic should remain constant over time-steps, so modulatory and recurrent connections are disabled. Secondly, the base logic only provides the optimal actions for L_0 (for L_1, optimal action is imitation of L_0), so neurons beyond the first output section are omitted. Thirdly, to ensure that agents can viably replicate the base logic phenotype, we limit the number of neurons in the hidden neuron section to half of that used in the agent networks (i.e. 8 instead of 16).

We evolve a base logic population of 225 networks for 2000 generations, and retain the best individual of the final generation as the base logic for the experiment. Then a population of 225 agent networks is evolved for 2,000,000 generations.

Evolving the agent population

In each generation, we split the population into 25 groups of 9 agents each. Within each group, every possible agent pairing interacts twice (once for each role assignment). At the start of each interaction, the L_0 agent generates a mental state, which remains constant throughout the interaction. Then the L_0 agent is exposed to every possible environmental state, in random order, while the L_1 agent tries to predict the L_0 agent's actions, seeing the shared environmental state but not the L_0 agent's mental state. After each action of the L_0 agent, the actual action choice is revealed to the L_1 agent (i.e. fed into its "partner action" input neurons), and weight update logic is performed. Each environmental state is seen only once per interaction, so simply remembering the partner's action choice is no viable strategy. For the L_0 agent, fitness payoff for an action is simply proportional to the action's proximity to the optimal action as given by the base logic. Below we use the performance of the L_1 agent on the last time-step of the interaction as a measure of the success of the learning process. However, L_1 fitness as used by the genetic algorithm is measured over all steps, so that a faster learner will have better fitness than a slow learner even if they perform equally well on the last time-step of the interaction.

At the end of each generation, agents are ranked per group, on basis of their L_0 performance and L_1 performance, with L_0 performance taking precedence over L_1 performance. That is, if agent X has a higher L_0 performance than agent Y, then X will be ranked above Y, independent of the L_1 performance scores. When X and Y have identical L_0 scores (a very common occurrence, especially once optimality on the L_0 task has been achieved), rank is decided by the L_1 score. This way of ranking avoids a trap in social evolution. If evolution co-opts the neural circuitry that determines L_0 behaviour for prediction of other agents' L_0 behaviour (i.e. L_1 behaviour), then when a mutant with better circuitry for L_0 behaviour appears, this mutant will have *worse* prediction ability with respect to the behaviour of its non-mutant peers (and those peers will have worse prediction ability with respect to the mutant). To prevent such effects from obstructing evolution of L_0 behaviour, L_0 performance should take precedence of L_1 behaviour. The best one third of each group (3 agents with the settings used here) overwrites the remaining two thirds with copies of themselves (so 2 copies per agent), to which mutation is applied. So effectively the parent agents each have 3 offspring, 2 mutated and 1 unmutated.

Results

Eight trials of the experiment described above were performed. Table 1 shows performance results for the final 1000 generations of each trial, and Figure 3 shows the evolution

process of a representative run. The scores displayed for L_0 are computed over individuals that have been retained without mutation from the previous generation (i.e. the unmutated offspring of the previous generation). Note that this implies that inclusion of an individual in the performance score is decided *before* its performance is measured. Such scores provide a good performance measure, as they are neither distorted much by mutation (as whole population averages are) nor by luck (as population best scores are). Scores displayed for L_1 are computed over all pairings between such individuals.

Run #	Max L_1 score without learning	L_0 score	L_1 score
1	.21	.9195	.9841
2	0	.9984	.9062
3	.20	.9861	.9641
4	.05	.9506	.8972
5	.34	.9199	.9745
6	0	.9961	.9690
7	.21	.9995	.9941
8	.29	.8839	.8974
average	.1625	.9568	.9483

Table 1. Performance scores. Scores are averages over the last 1000 generations of each run. Score averages for L_0 are taken over all individuals that have been copied from the previous generation without mutation. Score averages for L_1 are taken over all pairings between such individuals.

Variation in partner L_0 behaviour will harm L_1 performance of even optimal agents. As such, perfect performance cannot be expected. To allow for assessment of the evolved learning ability, we calculated the maximal expected fitness score agents without learning could obtain for each run's base logic, and show these alongside the performance scores in Table 1. These are the expected L_1 scores achieved by a hypothetical non-learning agent that for each environmental state simply picks the action that (over all possible mental states for an optimal L_0 partner agent) yields the best expected score. Note that the expected score for random behaviour is 0. We can see that L_1 performance in all runs widely exceeds the computed maxima for non-learning agents, showing that learning ability was indeed evolved. Average performance over all runs is well over 90% of the theoretical maximum, and some runs get very close to maximum performance (runs 3, 6 & 7).

These results indicate that the model, while far from perfect, is capable of producing agents that can learn how their interaction partner maps environmental states to actions. In that mapping, the partner's mental state plays a central role. As such, it seems that by observing their partner's behaviour, the agents in some form or another get a grasp on their partner's mental state. This suggests that these agents have evolved a primitive form of first-order Theory of Mind.

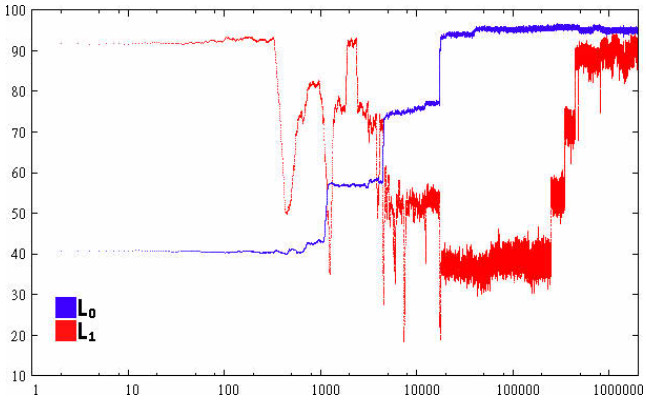

Figure 3. Evolution process of example run (run 4 in Table 1). As in Table 1, L_0 scores are averages over unmutated individuals and L_1 scores are averages over interactions between such individuals. Data points are smoothed over 100 generations. X-axis in log-scale. Initially the population adopted a simple L_0 behaviour, easy to imitate (hence high L_1 scores early on) but low in fitness. As the L_0 behaviour improves we see the L_1 performance fall, and then climb back up again as the learning ability necessary for prediction of the more complex L_0 behaviour evolves. Eventually the system settles in a state with high performance for both L_0 and L_1.

Future Work

The networks evolved here act as though they have first order ToM, but we have yet to establish how the mental state of partner agents is represented in the networks' activation patterns and/or weight modifications. Our primary future goal is to investigate this. It is well known that more often than not, networks evolved or trained to solve a given task represent their knowledge in highly diffuse and distributed fashion. However, as we have shown elsewhere (Arnold et al., 2012, 2013), when evolution and learning are combined, interactions between them tend to give rise to more organized forms of representation. In the present work too, we have a combination of learning and evolution at work.

Secondly, specific to ToM, we will investigate whether, in predicting an L_0 partner's behaviour, these nets use the circuitry they use when they themselves are performing at L_0. This would constitute a mirror-neuron-like, "placing oneself in another's shoes" approach to the problem.

Beyond the above, we aim to extend this research in the following directions: 1) More complex scenarios for the L_1 response (i.e. not mere prediction, but acting in anticipation of the L_0 agent's action). 2) Extension to higher (recursive) orders of ToM (by introducing endogenously controlled propagation looping in the NN architecture). 3) Once the model works for mental and environmental states of sufficient size, replacement of the randomly generated base logic with simple games or cognitive psychology experiments that involve ToM.

References

Arnold, S., Suzuki R. and Arita, T. (2012). Second Order Learning and the Evolution of Mental Representation. In *Artificial Life XIII: Proceedings of the Tenth International Conference on Artificial Life*, pp. 301–308. MIT press, Cambridge, MA.

Arnold, S., Suzuki, R., and Arita, T. (2013). Selection for Reinforcement-Free Learning Ability as an Organizing Factor in the Evolution of Cognition. *Advances in Artificial Intelligence*, vol. 2013, Article ID 841646, 13 pages.

Dennett, D. C. (1987). The Intentional Stance. MIT Press / Bradford Books, Cambridge, MA.

Gallese, V. (2007). Before and below 'theory of mind': embodied simulation and the neural correlates of social cognition. *Phil. Trans. R. Soc. B* 362, 659–669. doi:10.1098/rstb.2006.2002

Noble, J., Hebbron, T., Van Der Horst, J., Mills, R., Powers, S. T. and Watson, R. (2010). Selection pressures for a theory-of-mind faculty in artificial agents. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*, pp. 615–615. MIT Press, Cambridge, MA.

Premack, D. G., Woodruff, G. (1978). Does the chimpanzee have a theory of mind?. *Behavioral and Brain Sciences* 1 (4): 515–526. doi:10.1017/S0140525X00076512.

Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P. and Floreano, D. (2008). Evolutionary Advantages of Neuromodulated Plasticity in Dynamic, Reward-based Scenarios. In *Artificial Life XI: Proceedings of the Tenth International Conference on Artificial Life*, pp. 569–576. MIT Press, Cambridge, MA.

Takano, R. and Arita, T. (2006). Asymmetry between even and odd levels of recursion in a theory of mind. In *Artificial Life X: Proceedings of the Tenth International Conference on Artificial Life*, pp 405–411. MIT Press, Cambridge, MA.