Evolution of Connections in SHRUTI Networks

Joe Townsend, Ed Keedwell and Antony Galton
College of Engineering, Mathematics and Physical Sciences
University of Exeter
North Park Road, Exeter
jt231@ex.ac.uk, E.C.Keedwell@ex.ac.uk, A.P.Galton@ex.ac.uk

Abstract

SHRUTI is a model of how predicate relations can be represented and reasoned upon using a network of spiking neurons, attempting to model the brain's ability to perform reasoning using as biologically plausible a means as possible. This paper extends the biological plausibility of the SHRUTI model by presenting a genotype representation of connections in a SHRUTI network using indirect encoding and showing that working networks represented in this way can be produced through an evolutionary process. A multi-objective algorithm is used to minimise the error and the number of weight changes that take place as a network learns.

1 Introduction

Neural-symbolic integration concerns the representation of symbolic information in neural networks [Bader and Hitzler, 2005; Hammer and Hitzler, 2007]. Two motivations of this field are to combine the interpretability of symbolic systems with the adaptability of neural networks, and to produce models that point towards how symbols may be represented in biological neural networks. SHRUTI is a neural-symbolic network which models reasoning in the brain in a way that is claimed to be *biologically plausible* [Shastri and Ajjanagadde, 1993; Shastri, 1999]. The developers of SHRUTI discus the idea that the prerequisite structure required to enable it to learn logical relations can be realised in a way which is itself biologically plausible [Wendelken and Shastri, 2003]. They suggest that a SHRUTI network could be developed by a genotypic representation of rules that direct its growth; this is known as *indirect encoding*. However, the representation of SHRUTI networks using indirect encoding has not been implemented to our knowledge. We show that developmental genomes for creating connections between neurons in SHRUTI networks can be produced through evolution using *artificial development* [Chavoya, 2009], a form of evolutionary computing which uses indirect encoding.

We also want the evolved networks to yield minimal error when presented with test questions and to learn to do so with as few weight updates as possible, thus reducing the workload of the learning algorithm. A multi-objective algorithm was therefore chosen to minimise both of these properties.

2 Background

2.1 SHRUTI

SHRUTI [Shastri and Ajjanagadde, 1993; Shastri, 1999] is a neural-symbolic model of reflexive reasoning which has a number of biologically plausible traits: spiking neurons [Kumar *et al.*, 2010], temporal coding [Kumar *et al.*, 2010], and Hebbian learning [Hebb, 1949]. This section only covers the fundamentals required for SHRUTI to represent quantifier-free predicate logic, perform reasoning and learn relations, but a more complete explanation of SHRUTI can be found in the literature.

SHRUTI takes a localist approach to knowledge representation in that each concept is represented by its own ensemble of spiking neurons. Although a more distributed representation in which each neuron participates in the representation of multiple concepts is a more popular view in neuroscience, the localist view has not been completely ruled out [Bowers, 2009]. The bare minimum SHRUTI requires to work is one neuron per concept, but even by localist standards this lacks biological plausibility, and ensembles are preferred.

Each node in figure 1 represents an ensemble of spiking neurons. A predicate in the logic program is represented by a cluster of these nodes (e.g. $Buy(x, y)$), which contains a role node for each argument (e.g. $buyer$ and $object$) and collector (labelled '+' and '-') and enabler (labelled '?') nodes which direct the path of inference. Separate from the predicate clusters are entity nodes that represent entities which can fulfil the roles of predicate arguments ($John$, $Mary$, $Paul$ and $book$). A dynamic binding between a predicate argument and an entity can be formed by firing the nodes representing them so that their spike trains are in synchrony with each other. This method of temporal coding enables SHRUTI to overcome the variable binding problem. A predicate instance is composed of a set of argument-entity bindings for that predicate. For example, in order to ask the network in figure 1 'Does Mary own the Book?', the predicate $Own(x, y)$ is instantiated as $Own(Mary, Book)$ by firing the *owner* and *object* nodes of *Own* in synchrony with *Mary* and *Book* respectively. Activation of a predicate's enabler triggers a search for the truth of the current predicate instance, and activation of the positive or negative collector nodes asserts the truth or falsity of that instance. If neither collector is activated within a fixed time window, then the truth is regarded as unknown.

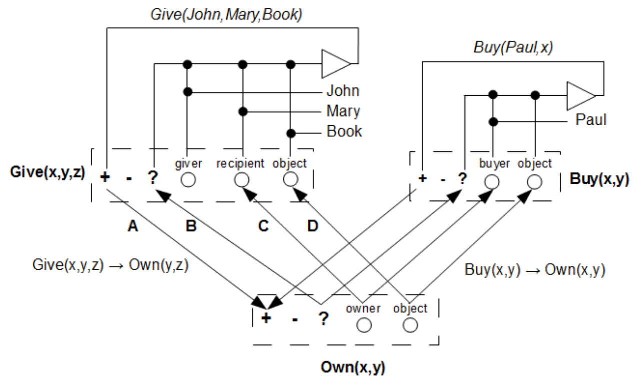


Figure 1: A simple SHRUTI network for the relations *Give(x,y,z)* $\rightarrow$ *Own(y,z)* and *Buy(x,y)* $\rightarrow$ *Own(x,y)*.

Connections between nodes represent relations between the predicates. Instantiating $Own(Mary, Book)$ as described above and firing the enabler of *Own* propagates the bindings to $Give(x, y, z)$ and $Buy(x, y)$ so that *recipient* and *buyer* are now bound to Mary, and the object nodes of each are bound to *Book*. The network is now asking 'Did somebody give Mary the Book?' ($Give(x, Mary, book)$) and 'Did Mary buy the book?' ($Buy(Mary, x)$). These bindings are then propagated further into sub-circuits representing long-term facts which can confirm or deny the truth of the corresponding predicate instance. Each such fact is composed of a set of inhibitory connections (shown as filled circles) and a fact node (shown as a triangle). The fact node only fires when the propagated set of dynamic bindings matches the static bindings encoded by the inhibitory connections. The facts in figure 1 state that John gave Mary the book ($Give(John, Mary, Book)$) and Paul bought something ($Buy(Paul, x)$). Because the dynamic bindings $Give(x, Mary, book)$ match the static bindings for the fact $Give(John, Mary, Book)$, the fact node is activated and in turn activates the positive collector of *Give*. This activation propagates to the positive collector of *Own* to assert that $Own(Mary, Book)$ is true.

Using Hebbian learning [Hebb, 1949], SHRUTI can learn relations between predicates [Wendelken and Shastri, 2003]. The training data takes the form of a sequence of events in the form of predicate instances that reflects causal relations between the predicates. If $P(a, b)$ is observed shortly before $Q(a, b)$, the weights of the connections supporting $P(x, y) \rightarrow Q(x, y)$ are strengthened according to equation 1 in order to reflect the likelihood that P is a cause of Q. However, if $Q(a, b)$ is not observed within a fixed time window of $P(a, b)$ occurring, the same weights are weakened according to equation 2 to reflect the likelihood that P is not a cause of Q. In both cases, the learning rate α is defined according to equation 3, which ensures that when a large amount of evidence has been found to support a relation it becomes more difficult to change its weights.

$$\omega_{t+1} = \omega_t + \alpha * (1 - \omega_t) \tag{1}$$

$$\omega_{t+1} = \omega_t - \alpha * \omega_t \tag{2}$$

$$\alpha = 1/NumberOfUpdates \tag{3}$$

New predicates can also be learned using recruitment learning [Feldman, 1982]. Nodes are divided into two groups: recruited and free. A free node becomes recruited when its connections to other nodes become strong enough, and when enough nodes have been recruited a new predicate is learned.

SHRUTI's learning model means that predicates and relations can be learned from a network of interconnected neurons. However, a fully interconnected network is impractical and lacks biological plausibility. The developers of SHRUTI argue that some pre-organisation of the network is required, and that this pre-organisation could be encoded in a biologically plausible way through genotypic representation [Wendelken and Shastri, 2003]. The idea of genome-instructed development of neural networks is possible through artificial development, but we have not found any literature that presents any attempts to implement the artificial development of SHRUTI networks.

2.2 Artificial Development

Artificial development is a form of evolutionary computing in which the genome encodes the phenotype indirectly by encoding a set of rules for its gradual development [Chavoya, 2009]. This is known as *indirect encoding*, the alternative of which is *direct encoding*, in which the genome encodes the structure of the phenotype explicitly. Indirect encoding is the more biologically plausible of the two, because DNA encodes instructions for the gradual development of an organism. Furthermore, indirect encoding has the advantage of scalability in that the size of the genotype is independent of that of the phenotype, in contrast to direct encoding. Such scalability is acheived through the compact representation of repeated substructures, and therefore indirect encoding is very suitable for the representation of SHRUTI networks, since each relation and fact is represented by a similar sub-circuit.

Artificial development has a number of applications, but the application of relevance to this paper is that of constructing neural networks. Some models for the artificial development of neural networks use graph grammars [Kitano, 1994; Gruau, 1994], which are adapted from Lindenmayer systems [Lindenmayer, 1968]. A number of other approaches to indirect encoding which do not involve graph grammars and exhibit more biological plausibility also exist [Eggenberger, 1997; Hotz *et al.*, 2003; Khan *et al.*, 2010]. In these models, the connections between neurons are often referred to by their biological counterpart of axons, and have positional attributes. A neuron's axons can grow in Euclidean grid space according to the genotypic instructions and connections between neurons are formed when axons meet [Eggenberger, 1997; Hotz *et al.*, 2003; Khan *et al.*, 2010].

3 Evolving SHRUTI Networks

This paper aims to demonstrate that simple SHRUTI networks can be produced through artificial development, supporting the claim of SHRUTI's developers that the prerequisite structure required to learn relations can be realised in a biologically plausible way. However only certain elements of the SHRUTI architecture have been accounted for. Though the current genome model supports the Hebbian learning of relations, it is not yet capable of developing networks that can produce new predicates through recruitment learning. In general, this genome assumes the pre-existence of neuron clusters representing facts and predicates and therefore does not develop neurons, only the connections between them. Furthermore, each node is implemented with only one neuron and these experiments have yet to be tested on SHRUTI models that represent nodes as ensembles of neurons.

3.1 The SHRUTI Genome

This section presents an existing genome model produced by the authors for developing connections in a SHRUTI network [Townsend *et al.*, 2012]. Figure 2 shows an example set of rules for developing a working SHRUTI network both as they appear in the genome and as they appear in the form of a decision tree, where each rule is represented by a path from the root node to a leaf node. Leaf nodes represent actions to be performed and all other nodes represent conditions necessary for that action to take place. The network is presented with a temporal sequence of predicate instances supporting a set of causal relations. At each time t, all predicate instances occurring at t are observed and any existing connection weights are updated according to SHRUTI's Hebbian learning algorithm. The rules in the genome are then assessed for each possible node pair and the actions of any satisfied rules are executed. These actions may be the addition of a connection with a specified weight, or the removal of an existing connection. Each node in a possible node pair has a different label in the genome. 'SELF' refers to the node for which an input connection is being considered, and the node from which the connection is being considered is labelled 'P_INPUT' (possible input) if the connection does not exist or 'E_INPUT' (existing input) if it does exist.

The genome is a string of elements that describes the structure of the decision tree. Each sub-string describes one node (condition or action) so that each node is referenced by the index of the corresponding sub-string. Each condition's sub-string contains an element for the attribute to be tested, the operator ($<$,$\leq$,$=$,$\geq$,$>$,$\neq$), the value to test that attribute against, and the indices of the action or condition to branch to when the current condition is evaluated as true or false. For example, the first sub-string in the genome represents condition 1, and can be interpreted as follows: if the activity of SELF is above 0.5, branch to condition (sub-string) 4, otherwise branch to condition 3. If an index of 0 is specified, then the search terminates. Genomes may also contain conditions or actions which although not currently expressed (e.g. condition 2 in figure 2), may come to be if an index gene is mutated to branch to it.

Attributes which can be tested in this model are the node's current level of activity, its type (role, enabler or collector), the total number of inputs, the remaining time before the time window for coactivation closes, and the firing delay, which corresponds to a node's current phase. Additional attributes may be tested for existing inputs: the weight and the number of updates (how many times a connection has been strengthened or weakened).

The genome in figure 2 contains two rules. Rule 1 removes redundant connections and rule 2 establishes connections between two active neurons of the same type. The genome was tested on a number of different event sequences, each supporting sets of relations of different sizes. Each network developed was tested with a set of 'true or false' questions in the form of predicate instances and was found to answer all of these questions correctly. Statistics for the networks developed from these event sequences are presented in table 1. Even though sets may contain the same number of relations and predicates (e.g. sets 5 to 7), the number of arguments in each predicate may differ, resulting in different statistics for these sets. Although the size of the genome is fixed, the size of each developed network is different, showing that the size of the genotype is independent of the size of the phenotype and that the genome is therefore scalable.

SELF.act	>	0.5	4	3	E_IN.nInputs	<	4	6	0
(1) Condition					(2) Condition				

E_IN.weight	<	0.1	6	0	P_IN.act	>	0.5	5	0
(3) Condition					(4) Condition				

P_IN.type	=	SELF.type	7	0	DEL		ADD	0.1
(5) Condition					(6) Action		(7) Action	

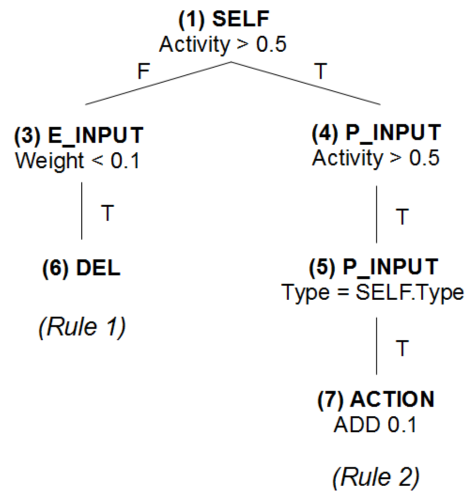


Figure 2: Rules for developing SHRUTI networks as they appear in the genome and as they appear in a decision tree.

Table 1: Number of connections and weight updates for networks produced by the genome in figure 2.

Set	Relations	Predicates	Connections	Updates
1	2	3	22	139
2	3	4	64	719
3	4	5	86	1056
4	4	7	53	535
5	5	6	65	721
6	5	6	80	852
7	5	6	83	1064
8	6	7	80	1168
9	6	7	73	730

3.2 Fitness Function

The aim of evolving SHRUTI networks was to minimise both the area beneath the error-time graph and the number of weight updates performed in training the network.

As the network develops, error will change during development as it learns more relations. Therefore the area beneath the error-time graph (which will henceforth be referred to as *e-area*) was chosen as an objective in order to encourage the algorithm to converge not only towards networks of minimum error but towards networks that can achieve minimum error as early as possible. To calculate an approximation of the e-area, error was measured at five intervals during the development of the network.

Error was measured as the difference between maximum accuracy and the accuracy obtained. Accuracy is based on how many questions the network is capable of answering correctly, and therefore maximum accuracy is achieved by answering all questions correctly. A question takes the form of a predicate instance, for example $P(a, b)$, which a developed network must assert as true, false, or unknown. However, rather than simply counting the number of correct answers, accuracy is a function on the number of correct collector activations. Each answer to a question consists of a positive and negative collector state, and therefore takes one of four values: [1,0] (true), [0,1] (false), [0,0] (unknown) and [1,1] (contradiction).

Answers are contained in a matrix a, where each row a_i represents one answer and contains an element for each collector activation. Matrix a is compared against a target matrix t. Each row a_i in the answer matrix is assigned a score and accuracy is measured as the sum of these scores as in equation 4. In the original scoring function S_1, a score of 1 was assigned to each correct collector activation as shown in equation 5. However, this resulted in high accuracies even when all questions were answered as unknown. All correct answers contain at least one inactive collector, and therefore by simply guessing [0,0] (unknown) for all questions the total accuracy could be a high percentage of the maximum accuracy as shown in table 2 (6 compared to a maximum accuracy of 8). Furthermore, this is higher than the accuracy obtained by guessing [1,1] for all questions, which may be considered 'better' from an evolutionary perspective since such networks are at least trying to answer questions. To overcome this problem, the score function was modified to create a new function S_2 as shown in equation 6, which gives a total accuracy of 0 to networks which always answer [0,0], and adds more weight to correct 'true' or 'false' questions by assigning them a score of 3 as opposed to just 2. Table 2 shows that by using S_2, networks only answering [0,0] are now assigned the absolute minimum accuracy, and that the accuracy of networks answering all questions correctly is even greater than before.

$$Accuracy(a) = \sum_{i=0}^{n} S(a_i) \tag{4}$$

$$S_1(a_i) = \sum_{j=0}^{m} 1 - |t_{i,j} - a_{i,j}| \tag{5}$$

$$S_2(a_i) = \begin{cases} 0 & \text{if } \sum_{i=0}^{n}\sum_{j=0}^{m} a_{i,j} = 0 \\ 3 & \text{if } \sum_{j=0}^{m} t_{i,j} = 1 \text{ and } S_1(a_i) = 2 \\ S_1(a_i) & \text{otherwise} \end{cases} \tag{6}$$

Table 2: A comparison of the outputs of two different scoring functions. Numbers given in bold denote total accuracies.

Expected answer		Score 1 (S_1)			Score 2 (S_2)		
+	-	All 0,0	All 1,1	All correct	All 0,0	All 1,1	All correct
1	0	1	1	2	0	1	3
0	1	1	1	2	0	1	3
0	0	2	0	2	0	0	2
0	0	2	0	2	0	0	2
		6	**2**	**8**	**0**	**2**	**10**

In addition to minimising e-area, the second objective was to minimise the number of weight updates performed on the network. Minimising this reduces the workload of the learning algorithm and constrains the number of connections in the network, because as the number of connections in a network increases, so does the number of the weights the algorithm has to update.

3.3 Evolutionary Algorithm

Given that this is a multi-objective problem, NSGA-II [Deb *et al.*, 2002] was chosen to evolve the networks. The algorithm was run with an initial population size of 100 over 500 generations, repeated over 50 trials. Genomes were fixed at a size of twelve conditions or actions and binary tournament selection with replacement was used to select genomes for recombination. Crossover was performed at a rate of 90% by randomly swapping a sub-tree from each parent. Children were then mutated by selecting sub-strings representing nodes with a probability of 10%, randomly choosing one of the elements from each and choosing new values according to a uniform distribution.

Fitness was calculated on a set of training questions and the performance of the final population was tested against a set of test questions. Set 7 from table 1 was chosen as the event sequence used for training and testing evolved networks. The set of training questions contained the minimum set of questions required to demonstrate that the desired relations had been learned correctly. In this case, the set contained three questions for which the expected answer was 'true', three expected to be 'false' and seven expected to be 'unknown', totalling thirteen questions and a maximum accuracy of 32 according to the scoring function. The test questions contained all other possible questions with the exception of predicate instances which were already encoded as facts, since these would always be answered correctly.

4 Results

4.1 Performance on Training Questions

Figure 3 shows samples obtained from 50 trials when minimising the e-area and the number of weight updates performed by a network during the learning process. Points marked with a dot in figure 3 indicate genomes which developed networks capable of answering all training questions correctly. 224 zero-error networks were found across 49 of the trials. In general, three very distinct groups of networks emerged, each of which was found to employ a different strategy for answering questions. These groups are indicated by the three boxes in figure 3. Each group's behaviour was analysed by sampling ten genomes from each.

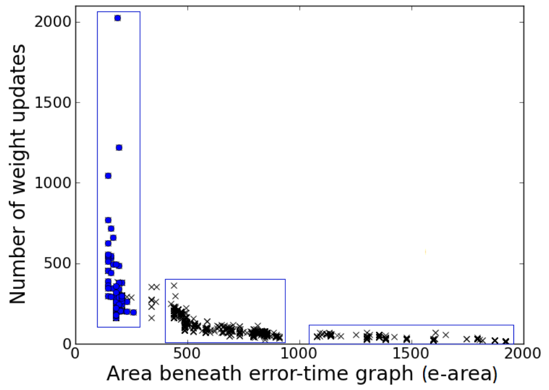


Figure 3: Points obtained from 50 trials on training data. Points marked with a dot represent genomes that answered all questions correctly.

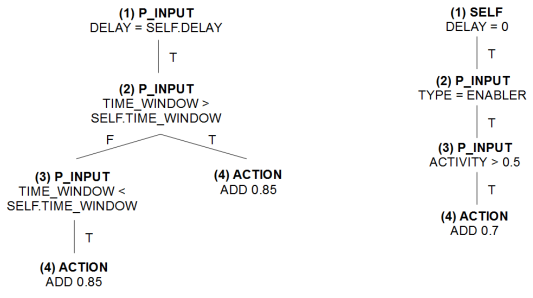
First group: $0 < $ e-area < 250

Errors in this group range between 0 and 2, though the error of the majority is 0. Genomes which construct these networks behave like rule 2 from the genome constructed in section 3.1, in that connections between SELF and P_INPUT are produced when both nodes are active and of the same type. However, equivalent conditions or combinations of conditions also emerged, rather than testing activity or type directly (figure 4(a)).

Where the number of updates is greater, at least one of the conditions that restrict the number of connections may be missing from the rules. For example, if the genome in figure 2 bypassed condition 4, i.e. did not require that P_INPUT was active in order for a connection to be created, a maximum accuracy network would still be developed. However the resulting network would contain a number of superfluous connections, therefore increasing the number of weight change operations the network has to perform.

Second group: $400 < $ e-area < 900

Errors in this group range between 5 and 15. However only one genome is found for errors of both 5 and 15, so the boundaries of any real interest are networks with errors of 6 and 14. Answers given by networks in this range depend only on the predicate queried without reference to its arguments. For example, the network might always answer 'true' for predicate P, 'false' for predicate Q and 'unknown' for predicate R. As error increases, questions for a greater number of predicates were always answered with 'unknown', from networks with an error of 6 and up to networks with an error of 14. 14-error networks always answer 'true' or 'false' for only one of the predicates and 'unknown' for everything else. Anything less will result in all questions being answered 'unknown' and the genome's performance will be penalised to the maximum error of 32. For all of these genomes, connections were only formed between enablers and collectors, but there was some restriction on the number of inputs that a node could have before a connection could be added. Figure 4(b)) shows a genome for constructing networks that yield an error of six.



(a) (b)

Figure 4: Example genomes for constructing networks from the first (a) and second (b) groups.

Third group: $1050 < $ e-area < 1900

Errors in this group range between 11 and 32, though the error of the majority is 32, i.e. the maximum possible error. In the maximum-error genomes in this cluster with the largest e-areas, any *add* actions they contained were never expressed. Therefore connections are never created, resulting in empty networks which always answer [0,0] (unknown) for every question at any time. As a result, the error and the e-area are always maximum whilst the number of weight updates is minimal. Maximum-error networks with a smaller e-area construct some connections and temporarily yield a non-maximal error, but remove these connections after a time and yield maximum error at the end of development.

4.2 Performance on Test Questions

The evolved genomes were tested by asking the developed networks a set of test questions which was larger than the set of training questions. Figure 5 shows how the networks performed on the test set. The general shapes of the Pareto fronts remain roughly the same for both sets. In particular, every genome capable of developing networks which answered all training questions correctly could also answer every test question correctly. Because most of the evolved networks perform as well in the test questions as they do on the training questions, these results demonstrate that the performance of evolved genomes is robust to unseen test questions.

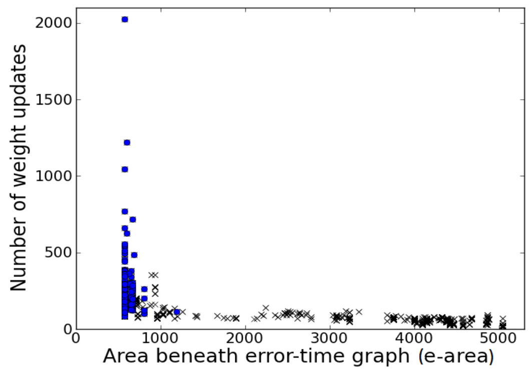


Figure 5: Points obtained by running genomes from 50 trials on test data.

5 Conclusions

Using NSGA-II, three groups of genomes for developing connections in SHRUTI networks emerged, each with its own distinct strategy for answering questions. One of these groups was successful in producing networks that behaved like regular SHRUTI networks, answering all questions correctly based on evidence learned from observed events. One claim of SHRUTI's developers was that reasoning can be the spontaneous and natural outcome of a system of neurons, and the findings in this paper support the idea that this itself can be the outcome of an evolutionary process. This supports another claim of the SHRUTI developers that the prerequisite structure required to enable the learning of relations in SHRUTI can be realised through a model of biological development, adding another dimension of biological plausibility to the model. However, the results only account for the basic SHRUTI model. The genome does not yet support nodes formed of multiple neurons, the creation of new neurons and the formation of new predicates through recruitment learning. To support the idea of a developmental SHRUTI model even further, we propose to investigate this.

References

[Bader and Hitzler, 2005] Sebastian Bader and Pascal Hitzler. *Dimensions of Neural-Symbolic Integration: A Structured Survey*, volume 1, pages 167–194. College Publications, London, 2005.

[Bowers, 2009] Jeffrey S Bowers. On the biological plausibility of grandmother cells: Implications for neural network theories in psychology and neuroscience. *Psychological Review*, 116(1), 2009.

[Chavoya, 2009] Arturo Chavoya. *Artificial Development*, volume 1 of *Studies in Computational Intelligence*, pages 185–215. Springer, 2009.

[Deb *et al.*, 2002] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[Eggenberger, 1997] Peter Eggenberger. Creation of neural networks based on developmental and evolutionary principles. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 337–342. Springer-Verlag, 1997.

[Feldman, 1982] Jerome A. Feldman. Dynamic connections in neural networks. *Biological Cybernetics*, 46:27–39, 1982.

[Gruau, 1994] Frédéric Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3(2):151–183, 1994.

[Hammer and Hitzler, 2007] Barbara Hammer and Pascal Hitzler. *Perspectives of Neural-Symbolic Integration*. Springer, Berlin, 2007.

[Hebb, 1949] Donald Olding Hebb. *The Organization of Behavior: A neuropsychological theory*. Wiley, New York, 1949.

[Hotz *et al.*, 2003] Peter Eggenberger Hotz, Gabriel Gómez, and Rolf Pfeifer. Evolving the morphology of a neural network for controlling a foveating retina - and its test on a real robot. In *Proceedings of The Eighth International symposium on artificial life*, pages 243–251, 2003.

[Khan *et al.*, 2010] Gul Muhammad Khan, Julian F. Miller, and David M. Halliday. *Intelligent agents capable of developing memory of their environment*, pages 77–114. UEFS, 2010.

[Kitano, 1994] Hiroaki Kitano. Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms. *Physica D: Nonlinear Phenomena*, 75(1-3):225–238, 1994.

[Kumar *et al.*, 2010] Arvind Kumar, Stefan Rotter, and Ad Aertsen. Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. *Nature Reviews Neuroscience*, 11(9), 2010.

[Lindenmayer, 1968] Aristid Lindenmayer. Mathematical models for cellular interactions in development. *Journal of Theoretical Biology*, 18(3):280–299, 1968.

[Shastri and Ajjanagadde, 1993] Lokendra Shastri and Venkat Ajjanagadde. From simple associations to systematic reasoning. *Behavioral and Brain Sciences*, 16(3):417–494, 1993.

[Shastri, 1999] L. Shastri. Advances in SHRUTI - a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence*, 11:79–108, 1999.

[Townsend *et al.*, 2012] Joe Townsend, Ed Keedwell, and Antony Galton. A scalable genome representation for neural-symbolic networks. Birmingham, 2012. Proceedings of the First Symposium on Nature Inspired Computing and Applications (NICA) at the AISB/IACAP World Congress 2012.

[Wendelken and Shastri, 2003] Carter Wendelken and Lokendra Shastri. Acquisition of concepts and causal rules in shruti. In *Proceedings of the Twenty Fifth Annual Conference of the Cognitive Science Society*, Boston, MA, 2003. Proceedings of the Twenty Fifth Annual Conference of the Cognitive Science Society.