A Scalable Genome Representation for Neural-Symbolic Networks

Joe Townsend, Antony Galton and Ed Keedwell
College of Engineering, Mathematics and Physical Sciences, Harrison Building, University of Exeter, North Park Road. Exeter, EX4 4QF
jt231@ex.ac.uk, A.P.Galton@ex.ac.uk , E.C.Keedwell@ex.ac.uk

Abstract. Neural networks that are capable of representing symbolic information such as logic programs are said to be neural-symbolic. Because the human mind is composed of interconnected neurons and is capable of storing and processing symbolic information, neural-symbolic networks contribute towards a model of human cognition. Given that natural evolution and development are capable of producing biological networks that are able to process logic, it may be possible to produce their artificial counterparts through evolutionary algorithms that have developmental properties. The first step towards this goal is to design a genome representation of a neural-symbolic network. This paper presents a genome that directs the growth of neural-symbolic networks constructed according to a model known as SHRUTI. The genome is successful in producing SHRUTI networks that learn to represent relations between logical predicates based on observations of sequences of predicate instances. A practical advantage of the genome is that its length is independent of the size of the network it encodes, because rather than explicitly encoding a network topology, it encodes a set of developmental rules. This approach to encoding structure in a genome also has biological grounding.

1 INTRODUCTION

Neural-Symbolic Integration [1, 9] is a field in which symbolic and sub-symbolic approaches to artificial intelligence are united by representing logic programs as neural networks or by developing methods of extracting knowledge from trained networks. The motivation behind this work is either the construction of effective reasoning systems, the understanding of knowledge encoded in neural networks, a model of human cognition, or a combination of these.

It may be possible to find powerful neural-symbolic networks through an evolutionary search. However, as the size of a logic program increases, so does the size of the network used to represent it. An evolutionary search for larger networks would take longer than it would for smaller networks as the search space would be larger, unless networks can be represented in a scalable way. *Artificial development* is a sub-field of evolutionary computing in which genomes encode rules for the gradual development of phenotypes rather than encoding their structures explicitly [3]. The genomes are scalable because genomes of equal length can produce solutions of different sizes. Among other applications, this encoding method can be applied to the representation of neural networks. This method of encoding networks is referred to as *indirect encoding*.

In addition to producing powerful reasoning systems, representing neural-symbolic networks in this way is more biologically plausible than encoding topologies directly. Because neural-symbolic networks claim to be a step towards a model of human cognition, it seems reasonable to develop them in a way which is also biologically plausible. If human cognition can be produced through evolution and development, then perhaps an artificial model of cognition can be produced through artificial models of evolution and development.

No attempt has yet been made to evolve neural-symbolic networks using artificial development. This paper introduces a scalable genome representation of neural-symbolic networks which adhere to a model known as SHRUTI [22, 23]. The genome was successful in its ability to construct four SHRUTI networks that were able to learn a set of relations between logical predicates. The intention is to eventually produce these genomes using an evolutionary algorithm, but this algorithm has yet to be implemented. Nonetheless, if SHRUTI networks can be produced through artificial development, it opens the possibility that other neural-symbolic models can be too. Section 2 provides an overview of SHRUTI and artificial development models used for the evolution of standard neural networks. Section 3 describes the experiments performed, the genome model used in these experiments and the target networks. Section 4 presents and discusses the results and section 5 concludes.

2 BACKGROUND

2.1 SHRUTI

SHRUTI is a neural-symbolic model in which predicates are represented as clusters of neurons and the relations between them as connections between those neurons [22, 23]. Predicate arguments are bound to entities filling the roles of those arguments by the synchronous firing of the neurons representing them. There is therefore no need to create a connection for every argument-entity combination. The SHRUTI authors claim that their approach, known as *temporal synchrony*, has biological grounding in that it is used for signal processing in biological neurons. SHRUTI can be used for forward or backward reasoning. In forward reasoning, the system is used to predict all the consequences of the facts. In backward reasoning, the system is used to confirm or deny the truth of a predicate instance based on the facts encoded. In other words, a backward reasoner answers ‘true or false’ questions. This paper concerns backward reasoning only.

Figure 1 provides an example of a basic SHRUTI network. Argument and entity neurons fire signals in phases, and a predicate is instantiated by firing its argument neurons in phase

with the entities fulfilling roles in the predicate instance. The other neurons in a predicate cluster fire signals of continuous phase only upon receipt of signals of the same nature. Positive and negative collectors (labelled '+' and '-') fire when the current predicate instance is found to be true or false respectively, and enablers (labelled '?') fire when the truth value of the current predicate instance is queried. Relations between predicates are established by linking corresponding neurons such that argument bindings are propagated between predicates. Facts are represented by static bindings between entities and predicate arguments such that the firing of a corresponding *fact neuron* (represented as a triangle in figure 1) is inhibited if the current dynamic bindings of the predicate do not match the static bindings of the fact.

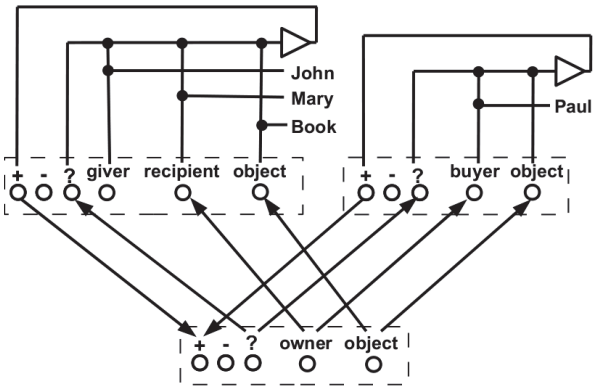


Figure 1 – A simple SHRUTI network for the relations Give(x,y,z) → Own(y,z) and Buy(x,y) → Own(x,y) and the facts Give(John, Mary, Book) and Buy(Paul,y)

The example network in figure 1 represents the relations *Give(x,y,z) → Own(y,z)* (if person x gives z to person y, then person y owns z) and *Buy(x,y) → Own(x,y)* (if person x buys y, then person x owns y). Also, two facts are represented: *give(John,Mary,Book*) (John gave Mary the book) and *buy(Paul,x)* (Paul bought something). This network is configured for backward reasoning. If one wishes to find the truth for *own(Mary, Book)* (Does Mary own the Book?), an instance of the *own* predicate must first be created by firing its *owner* and *object* neurons in the same phases as the neurons representing *Mary* and *Book* respectively. This creates a pair of dynamic bindings. The enabler (?) of *own* must also be fired to indicate that a search of *own*'s current instance is sought. The dynamic bindings are propagated along the connections to *give* and *buy* such that the neurons representing *recipient* and *buyer* are now firing in phase with *Mary* and the neurons representing *object* for *give* and *buy* are firing in phase with *Book*. *Give* and *buy* are therefore instantiated with the queries *give(x, Mary, Book)* (did somebody give Mary the book?) and *buy(Mary, Book)* (did Mary buy the book?). The dynamic bindings are then propagated to the static bindings representing facts. The static bindings of *buy(Paul,x)* do not match the dynamic bindings of *buy(Mary, Book)*, and so the static bindings inhibit the firing of the corresponding fact node which would otherwise activate the positive collector of *buy*. However, the dynamic bindings of *give(x, Mary, Book)* do match the static bindings of *give(John,Mary,Book*). The corresponding fact node is therefore not inhibited and activates the positive collector of *give* to assert that *give(x, Mary, Book)* is true. This collector in turn activates the positive collector of *own* to assert that *own(Mary, Book)* is also true, i.e. that Mary does indeed own the book.

There are many more features which may be included in a SHRUTI network. The literature also presents means of restricting dynamic bindings by entity types, conjoining predicates, enabling multiple instantiations of a predicate, and many other features. More complex models even use multiple neurons to represent one argument or entity, as the use of only one neuron to represent a concept lacks biological plausibility. One particular feature worth discussing in further detail is SHRUTI's learning mechanism [26] since it plays an important role in the developmental process discussed later in this paper.

SHRUTI's learning mechanism takes inspiration from Hebbian learning [10]. The training data is a sequence of events (predicate instances) observed over time that reflect the causal relations between the predicates. When two predicates are observed within a fixed time window, any connections representing relations between them are strengthened to increase the likelihood that the predicate observed first is a cause of the second. After a predicate is observed, any predicates that are connected to it but are not observed within the time window have those connections weakened to reflect the likelihood that they are not consequents of the first predicate. When a weight ω is strengthened, it is updated according to equation 1. When ω is weakened, it is updated according to equation 2. In both cases, the learning rate α is defined according to equation 3. This ensures that it becomes more difficult to change a relation for which evidence has been observed a large number of times.

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

$$\omega_{t+1} = \omega_t - \alpha * \omega_t \quad (2)$$

$$\alpha = \frac{1}{Number\ of\ Updates} \quad (3)$$

For example, if B(a,b) is observed shortly after A(a,b), connections will be updated to reflect A(x,y) → B(x,y). However, if A(a,b) is observed with no immediate observation of B(a,b), the same connection weights are weakened to reflect the lack of a relation between the two predicates. A new predicate can be recruited into the network once the connection weights of its neurons have gained sufficient strength.

The SHRUTI developers argue that some level of pre-organisation would be necessary for this learning model to work, and that this pre-organisation could be the product of evolutionary and developmental processes. To support the biological plausibility of pre-organisation, they point to the work of Marcus [16], who proposed ideas similar to those found in artificial development. However, a further review of literature has failed to find any attempts to produce SHRUTI networks using artificial development or similar methods. This is what motivates the ideas proposed in this paper.

2.2 Artificial Development of Neural Networks

Artificial development is a form of evolutionary computing in which the genome encodes instructions for the gradual development of the phenotype. This method is argued to be more biologically plausible than the alternative of encoding the structure of a phenotype explicitly in the genome, as it is closer

to the means by which DNA encodes biological structures. Dawkins argues that DNA is not a blueprint of biological structure but is more like a recipe for its construction [5]. Artificial development can be applied to a range of problems, and Chavoya provides a recent overview of artificial development models [3]. However, this paper is only concerned with the artificial development of neural networks.

One approach to evolving neural networks involves genomes which encode network topologies [24, 25]. For example, the genome may contain a list of neurons and another list of connections between them, or it may represent a connection matrix. Such methods of encoding are often referred to as *direct encoding*. The disadvantage of direct encoding is that the size of the genome increases in proportion to that of the network it represents. The alternative, *indirect encoding*, overcomes this problem by encoding a set of rules for the gradual development of the network. Just as a biological organism's cells all contain the same DNA, neurons within networks encoded by indirect encoding all contain or refer to a copy of the same genome, which represents a set of developmental rules. These rules provide instructions as to how the neuron should develop, for example by duplicating or deleting itself or by establishing a connection to another neuron. The developmental process often begins with only one neuron. When a neuron divides, its genome is passed on to both of its children, which is how all neurons are able to share the same genome. Which developmental operation takes place depends on the current attributes of the neuron. Therefore even though all neurons share the same genome, which developmental operations are executed at which point in time will not necessarily be the same for each neuron. Some models for the artificial development of neural networks use graph grammars inspired by Lindenmayer systems [15], whereas others use more biologically inspired ideas where neurons and connections are defined in a two or three dimensional Euclidean grid space.

Figure 2 presents grammar trees used by Gruau to define cell division processes [8]. Each node in the tree describes a cell (neuron) division. The children of each node describe the following division for each child neuron produced by the previous division. Separate grammar trees define sub-trees, the roots of which can be referenced by leaf nodes of the main tree. A sub-structure can therefore be encoded once but reused multiple times. As a consequence, genomes are more compact and convergence speed during evolution is reduced because the search space is smaller. Kitano used grammar encoding to develop connection matrices [12, 13]. This method could also produce repeated sub-structures, evident from repeated patterns in the connection matrices produced.

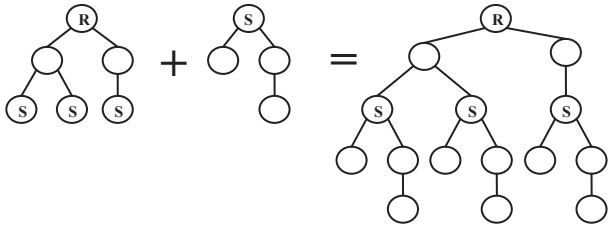


Figure 2 – Gruau's grammar trees. Each node corresponds to a cell division. The left-most tree describes the initial divisions from the root, and the second tree describes sub-trees which may grow from the leaves of the first tree. The third tree depicts the overall cell division process.

In the more biologically inspired methods, neurons and their connections (often regarded as *axons*, as with actual biological neurons) have positional attributes. A neuron's axon grows in the grid space, guided by developmental rules encoded in the genome, and form a connection when they come into contact with another neuron. The positional information of a neuron and its axons can be used to influence development. Eggenberger employed this idea using gene regulation [6, 7]. The activation of one gene, in addition to producing or deleting cells or connections, may also activate or inhibit the activation of other genes in the genome. Information can be passed between cells and the between the genes in those cells using artificial molecules. Concentration gradients of these molecules provide the positional information required to direct growth. Kitano also developed a similar model [14]. A different approach has been to use Cartesian genetic programs [18] to influence development in a grid [11]. The genome represents a set of seven interconnected programs. Three of these programs control signal processing in neurons, three control life-cycle processes such as the addition and deletion of neurons, and another controls weight updates.

Nolfi and Parisi used a means of measuring the fitness of a developing network that may prove useful in further research [19, 20]. Rather than simply measuring fitness at the end of the life-cycle of each phenotype, fitness was measured at different stages throughout development in order to observe how fitness increased over time. Such information on how the phenotype develops may be useful in the calculation of an overall fitness. For example, one might wish to measure overall fitness as the area under the fitness-time graph.

3 METHOD

A scalable genome model for the development of SHRUTI networks was produced, and the aim of the experiments conducted was to demonstrate, using an instance of this genome model, that the model could be used to develop four networks that could learn a set of logical relations between predicates based on a series of observed events. Each event was a predicate instance and each event sequence was representative of the relationships between the predicates in each logic program. Experiments were later repeated with shuffled event sequences in order to observe whether or not the genome could still develop networks which could represent the same logic programs. In additional experiments, sections of the genome were removed in order to see how the structures of the developed networks were affected. This section outlines how a SHRUTI model was implemented for these experiments, the genome model used to represent this implementation, and the target networks that were developed by the genome.

3.1 SHRUTI implementation

It seems reasonable to attempt the artificial development of a simple SHRUTI network before the development of more complex features is attempted. Therefore the basic SHRUTI model capable of learning as described in section 2.1 was implemented but more complicated features such as type restrictions and conjunction were excluded.

A minor adjustment was made to the learning mechanism in order to overcome difficulties learning certain structures. If two relations exist which share the same antecedent but with different signs for that antecedent, the system struggles to learn

both relations because the strengthening of one weakens the other unless both relations have been observed a sufficient number of times. To explain why this occurs, the learning of the relations P(x,y) → Q(x,y) and ¬P(x,y) → R(x,y) will be used as an example. The collectors of Q and R receive input from different collectors of P (+P and -P respectively). However, the enablers of Q and R both provide input to the same (and only) enabler of P (?P). Observation of P(x,y) and Q(x,y) within the time window will strengthen the connection from +P to +Q and the connection from ?Q to ?P. However, since ?P is activated and ?R is not, the connection from ?R to ?P will weaken. Likewise, if -P(x,y) and R(x,y) are observed within the time window, The connection from ?R to ?P will strengthen but the connection from ?Q to ?P will weaken.

To overcome this problem, the learning mechanism was configured by adjusting the learning rate α to update by a greater magnitude when strengthening weights than when weakening them. Therefore when weakening weights, α is defined as in equation 3, but when strengthening weights it is increased as shown in equation 4:

$$\alpha = \frac{1.25}{Number\ of\ Updates} \quad (4)$$

This makes it possible to learn these conflicting pairs of relations as long as the events that reflect them occur a sufficient number of times.

3.2 The Genome

In this first genome model, only the connections between neurons are developed, and not the neurons themselves. This approach assumes the pre-existence of neuron clusters representing facts and predicates, but there is room in future work to attempt the development of these clusters also.

Each genome describes a tree structure in which leaf nodes represent actions to be performed and all other nodes represent conditions. Each path through the tree structure from the root node to a leaf node represents a different rule. After each event has been observed and weights have been updated accordingly, the conditions encoded in a genome are tested for each neuron and each of its existing and possible inputs. If a leaf node is reached, the action it encodes is executed. The genome labels the current neuron for which input connections are being made as SELF. The neuron from which a connection is being considered is labelled as P_INPUT (possible input) if it does not yet exist and E_INPUT (existing input) if it does exist.

Figure 3 shows a set of conditions encoded by a genome for the development of a SHRUTI network and figure 4 shows them as a decision tree. Figure 5 presents an example of how an input connection is created using rule 2. To reduce execution time, conditions which affect SELF are considered first, so that evaluation of existing or potential inputs is only performed when necessary. Branching from one condition to another is therefore limited such that SELF conditions can branch to P_INPUT and E_INPUT conditions, but P_INPUT and E_INPUT conditions cannot branch to SELF conditions. The genome begins with a header containing the genome index of each type of condition and of the actions.

For each condition, the genome encodes the attribute to be tested, an operator (<,≤,=≥,>,≠), and the value to test that attribute against. Attributes which can be tested in this model are the neuron's current level of activity, its type (role, enabler or collector), the total number of inputs, and for existing inputs, the weight and the number of updates (how many times a connection has been strengthened or weakened). The genome also specifies the next condition to test or action to perform in the event of the current condition being evaluated as true or false. Alternatively, the tree search can end when a condition is evaluated as false and no actions are performed. For each action, the genome specifies one of two types of action to be performed: the addition or deletion of a connection. If a new connection is to be created, the genome also specifies the weight of the new connection.

SELF conditions:
1. If activity > 0.5, go to 2, else go to 5
2. If type = role node, go to 8, else go to 3
3. If type = enabler, go to 9, else go to 4
4. If type = collector, go to 10, else end.

E_INPUT conditions:
5. If number of updates > 7, go to 6, else end
6. If weight < 0.5, go to 12, else end.

P_INPUT conditions:
7. If activity > 0.5, go to 11, else end.
8. If type = role node, go to 7, else end.
9. If type = enabler, go to 7, else end.
10. If type = collector, go to 7, else end.

Actions:
11. Add connection with weight 0.1
12. Delete connection

Figure 3 – Conditions represented in the genome.

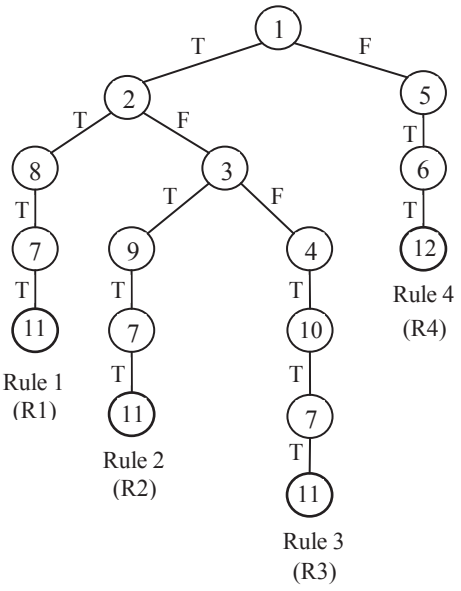


Figure 4 – A decision tree representation of the conditions given in figure 3 (T = True, F = False).

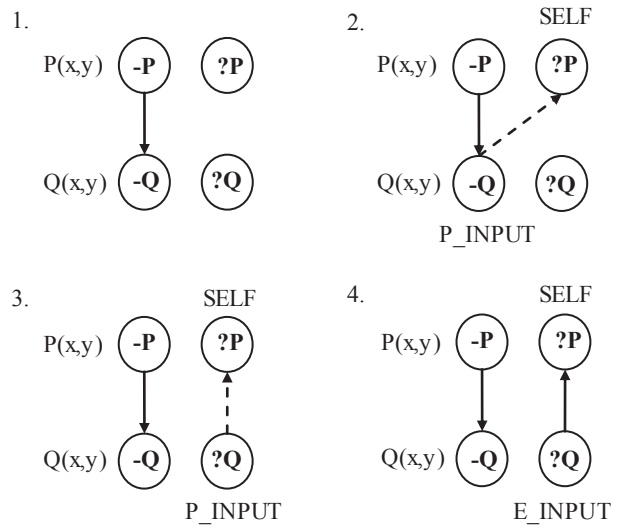


Figure 5 – How rule 2 is used to develop connections between enablers for the rule P(x,y) → Q(x,y). All nodes are active and a connection from -P to -Q already exists (1). The genome in ?P searches for input connections, starting with -Q (2). Both neurons are active and ?P is an enabler, but -Q is not, so no connection is made. The test is repeated for ?Q (3). Both neurons are enablers and both nodes are active, so an input connection is made (4).

In the genome defined in figures 3 and 4, rule 1 (R1) establishes connections between active role nodes. Rule 2 (R2) does the same for enablers and rule 3 (R3) does the same for collectors. Unwanted connections between neurons will inevitably form, but after Hebbian learning has taken place, their weights will weaken. Rule 4 (R4) prunes connections that are weak despite a large number of updates. A threshold of seven updates was chosen because this was the minimum value required to enable all test networks to learn desired connection weights without those connections being removed too early. This genome is one of a number which may be defined using this model to produce working SHRUTI networks.

3.3 Target networks

Figure 6 shows four target networks to be developed using the genome in figure 3. One of the networks is smaller than others in order to demonstrate the scalability of the genome. The two larger networks are of similar sizes but differ in structure. The logic program represented by the network with the label *SubNets* contains a relation and a predicate that are disjoint from the other relations and from each other. They are therefore each represented by separate sub-networks.

The initial state of each network is a set of neurons encoding facts connected to predicates, with the intention that connections will develop between predicate neurons over time in order to represent the relations between the predicates. For each network, a sequence of events in the form of predicate instances is defined. Each sequence reflects the relations between the predicates in the corresponding logic program. Different sub-sequences provide evidence for different sets of transitive relations. For example, observing the sub-sequence P(a,b), Q(a,b), R(a,b) supports the transitive pair of relations P(x,y)→Q(x,y), Q(x,y)→R(x,y).

Small

Expected Relations	Facts
P(x,y) → Q(x,y)	P(a,b)
¬P(x,y) → R(x,y)	¬P(c,d)

Large1

Expected Relations	Facts
P(x,y) → Q(x,y)	P(a,b)
Q(x,y) → ¬R(x,y)	¬Q(c,d)
¬Q(x,y) → S(x,y)	Q(e,f)
¬R(x,y) → ¬T(x,y)	S(g,h)
¬R(x,y) → ¬U(x,y)	
S(x,y) → V(x,y)	

Large2

Expected Relations	Facts
P(x,y) → ¬Q(x,y)	P(a,b)
¬P(x,y) → R(x,y)	¬Q(c,d)
¬Q(x,y) → ¬S(x,y)	¬P(e,f)
R(x,y) → T(x,y)	R(g,h)
¬R(x,y) → ¬U(x,y)	¬R(i,j)

SubNets

Expected Relations	Facts
P(x,y) → Q(x,y)	P(a,b)
R(x,y) → S(x,y)	R(c,d)
¬R(x,y) → T(x,y)	¬R(e,f)
¬T(x,y) → ¬U(x,y)	¬T(g,h)
	V(a,b)

Figure 6 – Target networks. Each table shows the relations which were expected to develop and the hard coded facts which make up the background knowledge. In the logic program represented by SubNets, the relation P(x,y) → Q(x,y) and the predicate V(a,b) are disjoint from the other relations and from each other.

Any number of events may occur at each time *t*, even zero. At each *t*, neurons that represent an observed predicate are fired and Hebbian learning is used to update the weights of the connections between the neurons that fire within a fixed time window of each other in order to build relations between predicates. Developed networks were tested by inputting 'true or false' questions and fitness was based on the number of questions answered correctly. The reader is reminded that each predicate includes two collectors: one positive and one negative, to assert the truth and falsity of the predicate respectively. Activation of one of these collectors shall be denoted 1, and deactivation 0. The truth of a predicate instance is therefore denoted by (1, 0), falsity by (0, 1), uncertainty by (0, 0) and contradiction by (1, 1). Fitness is measured as the number of correct collector activations. For example, consider a question with expected answer (1, 0). Answering (1, 0) would add 2 to the fitness, answering (0, 0) or (1, 1) would add 1, and (0, 1) would add 0.

For each network, the following statistics were recorded: the number of connection additions and deletions, the final number of connections and the final number of live connections. A connection is live when its weight is above the threshold of the neuron for which it is an input. Connections that are not live

make no contribution to inference in the network. However, this does not necessarily mean that all live connections do.

4 RESULTS

All test networks developed such that they could answer all of their test questions correctly. The same genome was successfully applied to the development of large and small networks, demonstrating that the genome is scalable in that its size is independent of the size of the phenotype. Further experiments observed how different components of the genome affected the network structure and what affected the change in fitness over time.

4.1 Network structure

Table 1 shows the statistics for each network. In each case, the total number of connections developed was not much greater than the number of live connections, meaning that only a few superfluous connections were developed.

Network	Connections	Live	Additions	Deletions
Small	10	10	30	20
Large1	49	36	138	89
Large2	43	35	86	43
SubNets	45	29	74	29

Table 1 – Statistics of fully developed networks: the total number of connections, the number of live connections (connections for which weight is greater than or equal to 0.5), and the number of connection additions and deletions.

Table 2 shows the results of removing different components of the genome when testing on the network *Large1*. In each case, maximum fitness was achieved with the same number of live connections. However, the total number of connections was greater because removing rules and conditions removed constraints on network size. The genome was constructed not only to develop networks capable of answering all questions correctly, but to do so with the minimal number of connections.

Excluded	Connections	Live	Additions	Deletions
None	49	36	138	89
Rule 4	98	36	98	0
Condition 7	171	36	491	320
Rule 4 and Condition 7	328	36	328	0

Table 2 – The effects of excluding rules and conditions from the genome when developing *Large1*.

Removing rule 4, which prunes superfluous connections, caused the total number of connections to double. However the number of additions decreased, implying that when rule 4 is included some of the connections it removes redevelop. Condition 7 limits connections to inputs from active neurons. Bypassing this caused an even greater increase in the number of connections. This, coupled with the tendency of deleted connections to redevelop, suggests that it is more beneficial to prevent the growth of superfluous connections than it is to delete them once created. Removing conditions 2 to 4 and 8 to 10, which limit connections to neurons of the same type, resulted in the network being unable to answer all questions correctly. Removing these conditions caused connections to form between enablers and collectors such that activation of a predicate's enabler triggered the immediate activation of one or both of its collectors, depending on which collectors were activated during training. All questions were therefore answered true (1,0), false (0,1) or both (1,1), but never unknown (0,0), and so questions for which unknown was the correct answer were answered incorrectly.

4.2 Fitness

Figure 7 shows the change in fitness as the network *Large1* develops, and figure 8 shows the change in fitness after shuffling the event sequence. Note that the initial fitness in both cases is not zero. This is due to the fact that fitness is based on the number of correct collector firings. An undeveloped network will answer all questions as 'unknown' (0, 0). For some test questions, this will in fact be the correct answer, so an undeveloped network automatically answers them correctly. For other questions, target answers are either (1, 0) (true) or (0, 1) (false), meaning that an answer of (0, 0) will be half correct for each of these questions. In summary, the initial fitness is due to the inability of an undeveloped network to fire any collectors and the large number of zeros (instances of collectors failing to fire) in the test data.

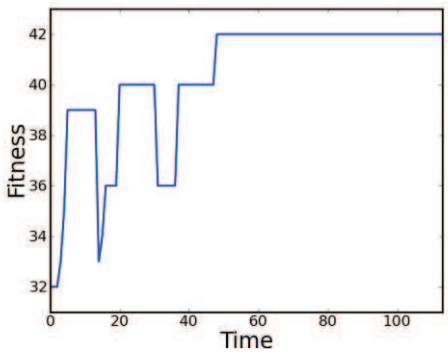


Figure 7 – The change in fitness over time for the network *Large 1*

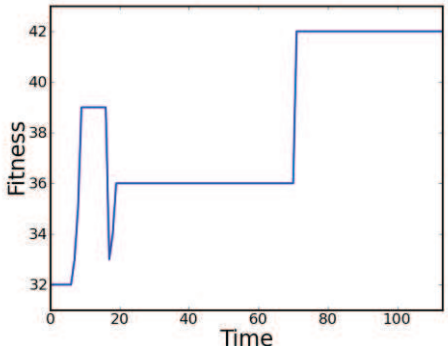


Figure 8 – The development of *Large1* after shuffling the event sequence.

In figures 7 and 8, maximum fitness is eventually achieved, but the fitness decreases and increases again before it reaches the maximum. This behaviour was caused by the weakening of some relations upon the strengthening of others, as described in section 3.1. In order to confirm that it was these conflicting relations that caused the trend of oscillating fitness, the learning experiment was repeated on a simple network which did not contain conflicting relations. The network represented a linear chain of predicates in which each predicate (with the exception of those at the beginning and end of the chain) was the consequent and antecedent of only one other predicate ($P(x,y) \rightarrow Q(x,y)$, $Q(x,y) \rightarrow R(x,y)$ …. $T(x,y) \rightarrow U(x,y)$). Figure 9 shows that in this case, the fitness only increased and never decreased, as no learned relations were disturbed by the learning of others.

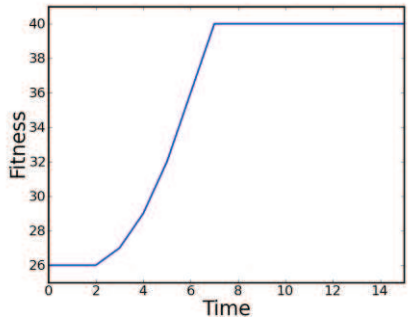
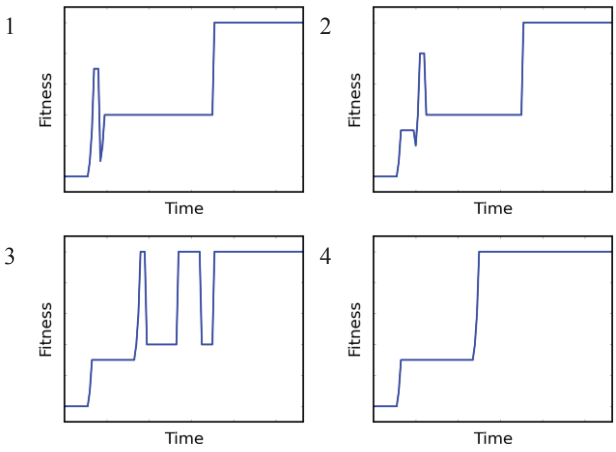


Figure 9 – The development of a network representing a linear chain of predicates

In the process of learning conflicting relationships which affect each other in this way, relations are learned and unlearned until both have been observed enough times (usually about 3) to support evidence for both, at which point both relations are successfully represented. This learning and unlearning of relations affects the truth values of predicate instances that depend on them, meaning that the correct assertion of these predicate instances is also periodic until the network settles. As a consequence, questions are periodically answered correctly and incorrectly before they can be consistently answered correctly, which explains the peaks and troughs in the graph. How soon a network settles into a state whereby this behaviour stops depends on the number of event sub-sequences supporting each relation and on the order in which they occur. This is due to the fact that the magnitude of change depends on the value of α, which is defined slightly differently for the weakening and strengthening of weights (as in equations 3 and 4 respectively) but is inversely proportional to the number of updates in both cases. In other words, how long the network takes to settle depends on how many times connections have been strengthened and how many times they have been weakened.

For example, consider the shuffled event sequence used for learning *Large1* as shown in figure 8. The set of relations dependent on $Q(x,y)$ is $[Q(x,y) \rightarrow \neg R(x,y), \neg R(x,y) \rightarrow \neg T(x,y), \neg R(x,y) \rightarrow \neg U(x,y)]$. The set of sub-sequences that supports this set will be referred to as X. The set of relations dependent on $\neg Q(x,y)$ is $[\neg Q(x,y) \rightarrow S(x,y), S(x,y) \rightarrow V(x,y)]$. The set of event sub-sequences supporting evidence for this set will be referred to as Y. The initial peak in fitness occurs when a member of X, X_1, completes at *t=7*, and drops again when Y_1 completes at *t=17*. Y_2 and Y_3 then complete at *t=25* and *t=39*. X_2 completes at *t=54*, but has no effect on fitness because the number of instances of X isn't enough to balance the connection weights. Y_4 completes at *t=66*. X_3 completes at *t=72* and instances of both X and Y have now occurred enough times that the relationships they each support are strong enough to maintain maximum fitness without the observation of one disturbing the relationship supported by the other. After X_3, further instances of X and Y occur interchangeably but fitness does not drop now that the relationships are balanced.

This hypothesis as to why the peaks and troughs occur was tested by moving X_1 further along the timeline of events in order to move the initial peak in fitness seen in figure 8 along the graph. This is demonstrated in figure 10. In the first image, X_1 is moved to occur just before Y_1, causing the peak to become narrower. In the second image, X_1 is moved to occur after Y_1 but just before Y_2, creating another narrow peak as X_1 improves fitness but Y_2 reduces it again. In the third image, X_1 is moved to occur just before Y_3 to temporarily increase fitness to the maximum. After Y_3 causes fitness to drop soon after, X_2 is able to increase it again, for a bit longer, before Y_4 causes a drop in fitness once more. After X_3, the network is balanced. In the final image, X_1 is moved to occur just before X_2 and this balances the relations. Note that unlike the other graphs, the third graph contains two peaks before fitness settles. In this case, instances of X and Y alternate more than they do in others, and the graph contains the greatest number of fitness peaks before the network settles. In the fourth graph, there are no peaks and only one change from Y to X before the network settles. Furthermore, the network settles slightly earlier than in the other trials. These behaviours imply that fitness peaks are caused by observation sub-sequences that alternate more, and that conflicting relations can be learned more quickly when the evidence for them alternates less.



1.	X_1	Y_1	Y_2	Y_3	X_2	Y_4	X_3	X_4	Y_5	X_5	Y_6	X_6
2.	Y_1	X_1	Y_2	Y_3	X_2	Y_4	X_3	X_4	Y_5	X_5	Y_6	X_6
3.	Y_1	Y_2	X_1	Y_3	X_2	Y_4	X_3	X_4	Y_5	X_5	Y_6	X_6
4.	Y_1	Y_2	Y_3	X_1	X_2	Y_4	X_3	X_4	Y_5	X_5	Y_6	X_6

Figure 10 – The effects of moving sub-sequence X_1, which supports evidence for relations depending on Q(x,y), further along the timeline. The resultant ordering of sub-sequences for each graph is displayed at the bottom.

This tendency of fitness to rise and fall before development is complete is similar to a phenomenon referred to as U-shaped development which has been observed in various aspects of cognitive development [2, 4, 17, 21]. One example from natural language is the way children learn past tense conjugation [21]. In early stages of natural language development children will know some regular and irregular past tense verbs and apply them correctly. However, data shows that once they realise that a large number of verbs are conjugated by the addition of 'ed' to those verbs, they over-generalise this rule to the irregular verbs as well, ignoring the irregularities they have already acquired and incorrectly conjugating them like any other. For example, upon noticing correct conjugations such as the derivation of 'reached' from 'reach' and 'heated' from 'heat', they often derive 'eated' from 'eat', 'goed' from 'go', and so forth, even though they correctly used 'ate' and 'went' before. Only once they have had more exposure to the English language and have heard regularities and irregularities frequently enough do they realise that the addition of 'ed' does not apply to all verbs. They are then able to apply regularities and irregularities correctly once again. In summary, the child's language ability gets worse before it improves. Though correction by adults may play some part in this process, it is largely credited to observations of how others use language. Errors tend to occur with verbs heard less often. Only when an irregularity is observed enough times is that irregularity able to 'block' the application of the rule to a verb stem. In SHRUTI's learning system, conflicting relations also require a sufficient number of observations before the representation of both relations can be balanced.

U-shaped development in children has been observed in a range of other cognitive tasks [2, 4, 17]. The U-shaped development observed in the SHRUTI learning model gives it another level of biological plausibility. Of course, the U-shaped development observed in SHRUTI is not caused by rules being over-generalised to irregularities but by rule pairs for which antecedents are of the same predicate but have different signs. However, the developmental process is similar in the sense that it is influenced by observations that may result in the ability of the developing structure declining before it is able to improve even further. It should also be noted that SHRUTI's U-shaped development is a result of the learning process and not of the developmental model which was implemented for these experiments. Nonetheless, the results above have been useful in determining that the genome model is able to support weights that continually gain and lose strength before they are consistently strong enough to represent relations. There was always the danger that a weight would lose enough strength that the genome would prune the connection before it was given a change to gain its strength back. This was not the case.

SHRUTI's U-shaped development will need to be taken into consideration when assessing the fitness of genomes in planned attempts to evolve them. A genome in the population which exhibits a lower fitness may have more potential for improvement than a genome with a slightly higher fitness. It may be necessary to adjust the measurement of fitness so that this potential is also taken into account, in addition to the number of questions a developed network can answer correctly. However, the challenge this idea presents is that of finding a way to quantify this potential.

5 CONCLUDING REMARKS

A scalable genome encoding of basic SHRUTI networks has been produced. A genome constructed using the presented model was successful in growing neural connections in SHRUTI networks such that those networks were able to correctly answer all their test questions correctly. Due to the rule-instructed growth, the size of the genome is independent of that of the phenotype, i.e. a network representing a logic program. The model applies the reuse of sub-structure as used by Gruau and Kitano. The four rules depicted in figure 4 share some repeated conditions, but these are only encoded once in the genome. Encoding these rules separately would have resulted in repeated encoding of these conditions, thus reducing the compactness of the genome.

The genome model proposed contributes towards two goals of neural-symbolic integration. For those interested in the practical application of neural-symbolic networks, a scalable means of representing them has been produced. With regards to developing a working model of logic representation in the human brain, this model is relevant because artificial development and neural-symbolic implementation both claim some degree of biological plausibility. The biologically plausible traits currently exhibited by the system as a whole include the indirect encoding and gradual development of the phenotype, the temporal synchrony and Hebbian learning employed by SHRUTI, and the U-shaped development observed in the change in fitness over time. However, one function that the current genome model lacks that would otherwise increase its biological plausibility further is the production of neurons. The current genome model only develops connections between neurons and not the neurons themselves. Biological development produces neurons and the genome model presented should eventually be updated to include rules for neuron production also. Cell division would be a suitable, biologically plausible means of implementing this.

The model presented is the first step towards producing SHRUTI networks, and possibly other types of neural-symbolic network, using artificial development. The next stage is to attempt the production of these genomes using an evolutionary algorithm. The current fitness function only takes into account the number of questions a network can answer correctly. However, it should be adapted to also take into account a network's potential to improve through further development. Successful evolution of these developmental genomes would contribute towards a means of producing artificial models of cognition that takes inspiration from the way cognitive structures emerge through natural evolution.

7 REFERENCES

[1] Bader, S., Hitzler, P (2005) "Dimensions of Neural-Symbolic Integration" in *We Will Show Them: Essays in Honour of Dov Gabbay*, **1**, College Publications, pp167-194.

[2] Baylor, A.L. (2001) "A U-shaped model for the development of intuition by level of expertise", *New Ideas in Psychology*, **19**, pp237-244.

[3] Chavoya, A. (2009), "Artificial Development", *Foundations of Computational Intelligence*, **1**, Springer, 185-215.

[4] Davis, J. (1997) "Drawing's Demise: U-shaped Development in Graphic Symbolization", *Studies in Art Education*, **38**, No 3, pp132-157.

[5] Dawkins, R. (1986), "The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design", Norton, New York.
[6] Eggenberger, P. (1997) "Creation of Neural Networks Based on Developmental and Evolutionary Principles" in *Proceedings of the International Conference on Artificial Neural Networks*, Springer-Verlag, pp337-342.
[7] Eggenberger, P., Gómez, G., Pfeifer, R. (2003) "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*, pp243-251.
[8] Gruau, F. (1994) "Automatic Definition of Modular Neural Networks", *Adaptive Behaviour*, **3**, No 2, pp152-183.
[9] Hammer, B., Hitzler, P (2007), "Perspectives of Neural-Symbolic Integration", Springer.
[10] Hebb, D.O. (1949) *"The Organization of Behaviour: A Neuropsychological Theory"*, Wiley.
[11] Khan, G.M., Miller, J.F., Halliday, D.M. (2010) "Intelligent Agents Capable of Developing Memory of Their Environment" in Loula A., Queiroz J., editors, *Advances in Modelling Adaptive and Cognitive Systems*, UEFS, pp77-114.
[12] Kitano, H (1990) "Designing Neural Networks Using Genetic Algorithms with Graph Generation System", *Complex Systems Journal*, **4**, pp461-476.
[13] Kitano, H. (1994) "Neurogenetic Learning: An Integrated Method of Designing and Training Neural Networks Using Genetic Algorithms", *Physica D: Nonlinear Phenomena*, **75**, No 1-3, pp225-238.
[14] Kitano, H. (1995), "A Simple Model of Neurogenesis and Cell Differentiation based on Evolutionary Large-Scale Chaos", *Artificial Life*, **2**, pp79-99.
[15] Lindenmayer, A. (1968) "Mathematical Models for Cellular Interactions in Development", *Journal of Theoretical Biology*, **18**, No 3, pp280-299.
[16] Marcus, G. (2001) "Plasticity and Nativism: Towards a Resolution of an Apparent Paradox" in *Wermter S., Austin J., Willshaw D., editors, Emergent Neural Computational Architectures Based on Neuroscience, Lecture Notes in Computer Science*, **2036**, Springer Berlin / Heidelberg, pp368-382.
[17] McNeil, N.M. (2007) "U-shaped Development in Math: 7-Year-Olds Outperform 9-Year-Olds on Equivalence Problems", *Developmental Psychology*, **43**, No 3, 687-695.
[18] Miller, J.F., Thomson P. (2000) "Cartesian Genetic Programming" in Proceedings of the 3rd European Conference on Genetic Programming, **1802**, Berlin, Springer-Verlag, pp121-132.
[19] Nolfi, S., Parisi, D. (1995) "Evolving Artificial Neural Networks That Develop in Time" in Morán F., Moreno A., Merelo J., Chacón P., editors, *Advances in Artificial Life, Lecture Notes in Computer Science*, **929**, Springer Berlin / Heidelberg, pp353-367.
[20] Nolfi, S., Parisi, D. (1996) "Learning to Adapt to Changing Environments in Evolving Neural Networks", *Adaptive Behaviour*, **5**, pp75-98
[21] Pinker, S. (1999) "Kids Say the Damedest Things" in *Words and Rules: The Ingredients of Language*, pp189-210.
[22] Shastri, L., Ajjanagadde, V. (1993) "From Simple Associations to Systematic Reasoning", *Behavioral and Brain Sciences*, **16**, No 3, pp417-494.
[23] Shastri, L. (1999) "Advances in SHRUTI – A Neurally Motivated Model of Relational Knowledge Representation and Rapid Inference using Temporal Synchrony", *Applied Intelligence*, **11**, pp79-108.
[24] Siebel, N.T., Sommer G. (2007) "Evolutionary Reinforcement Learning of Artificial Neural Networks", *International Journal of Hybrid Intelligent Systems*, **4**, No 3, pp171-183
[25] Stanley, K.O., Miikkulainen R. (2002), "Efficient Evolution Of Neural Network Topologies" in *Proceedings of the Genetic and Evolutionary Computation Conference,* Morgan Kaufmann, pp1757-1762.
[26] Wendelken, C., Shastri, L. (2003) "Acquisition of Concepts and Causal Rules in SHRUTI" in *Proceedings of the Twenty Fifth Annual Conference of the Cognitive Science Society*, Boston.