University of Exeter
Department of Computer Science

Artificial Development of Neural-Symbolic Networks

Joseph Paul Townsend

March 2014

Supervised by Dr. Ed Keedwell and Dr. Antony Galton

Submitted by Joseph Paul Townsend, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science , March 2014.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

(signature) .....

Abstract

Artificial neural networks (ANNs) and logic programs have both been suggested as means of modelling human cognition. While ANNs are adaptable and relatively noise resistant, the information they represent is distributed across various neurons and is therefore difficult to interpret. On the contrary, symbolic systems such as logic programs are interpretable but less adaptable. Human cognition is performed in a network of biological neurons and yet is capable of representing symbols, and therefore an ideal model would combine the strengths of the two approaches. This is the goal of *Neural-Symbolic Integration* [4, 16, 21, 40], in which ANNs are used to produce interpretable, adaptable representations of logic programs and other symbolic models.

One neural-symbolic model of reasoning is *SHRUTI* [89, 95], argued to exhibit biological plausibility in that it captures some aspects of real biological processes. SHRUTI's original developers also suggest that further biological plausibility can be ascribed to the fact that SHRUTI networks can be represented by a model of genetic development [96, 120]. The aims of this thesis are to support the claims of SHRUTI's developers by producing the first such genetic representation for SHRUTI networks and to explore biological plausibility further by investigating the evolvability of the proposed SHRUTI genome.

The SHRUTI genome is developed and evolved using principles from *Generative and Developmental Systems* and *Artificial Development* [13, 105], in which genomes use *indirect encoding* to provide a set of instructions for the gradual development of the phenotype just as DNA does for biological organisms. This thesis presents genomes that develop SHRUTI representations of logical relations and episodic facts so that they are able to correctly answer questions on the knowledge they represent.

The evolvability of the SHRUTI genomes is limited in that an evolutionary search was able to discover genomes for simple relational structures that did not include conjunction, but could not discover structures that enabled conjunctive relations or episodic facts to be learned. Experiments were performed to understand the SHRUTI fitness landscape and demonstrated that this landscape is unsuitable for navigation using an evolutionary search. Complex SHRUTI structures require that necessary substructures must be discovered in unison and not individually in order to yield a positive change in objective fitness that informs the evolutionary search of their discovery.

The requirement for multiple substructures to be in place before fitness can be improved is probably owed to the localist representation of concepts and relations in SHRUTI. Therefore this thesis concludes by making a case for switching to more distributed representations as a possible means of improving evolvability in the future.

Acknowledgements

I would like to thank my supervisors Dr. Ed Keedwell and Dr. Antony Galton for all the help they have provided throughout my research and for encouraging me to share my work with the wider research community through various publications. Furthermore, I would like to thank them for taking an interest in my research in the first place and for their willingness to supervise it.

I would also like to thank my parents, Derek William and Margaret Townsend, and my brother Martyn Townsend, for supporting, advising and encouraging every single decision I have made to date regarding my education, work and life in general.

Finally, I would like to acknowledge my friends James Gould, David Singeisen, Marjorie Gehrhardt and Rachel Warren. Over the past few years we have shared the joys and hardships of academia together and supported each other in doing so. I would like to thank these people for their support, particularly for encouraging me to keep going. I would like to encourage them to do the same and wish them all the best in their future work, be it in academia or elsewhere.

Contents

List of Tables

List of Figures

Publications

Some of the background discussion from chapter 2 and some preliminary results from chapter 3 have been published in:

> J. Townsend, E. Keedwell, and A. Galton. Artificial development of biologically plausible neural-symbolic networks. *Cognitive Computation*, 6(1):18-34, March 2014.

Chapter 3 extends work previously published in:

> J. Townsend, E. Keedwell, and A. Galton. A scalable genome representation for neural-symbolic networks. In *Proceedings of the First Symposium on Nature Inspired Computing and Applications (NICA) at the AISB/IACAP World Congress 2012*, Birmingham, July 2012.
>
> J. Townsend, E. Keedwell, and A. Galton. Artificial development of connections in SHRUTI networks using a multi objective genetic algorithm. *Proceedings of the fifteenth annual conference on genetic and evolutionary computation conference companion*. ACM, Amsterdam, July 2013.
>
> J. Townsend, E. Keedwell, and A. Galton. Evolution of Connections in SHRUTI Networks. *Proceedings of the 9th International Workshop on Neural-Symbolic Learning and Reasoning NeSy13*, Beijing, August 2013

Definitions

Accuracy	A measure of the efficacy of a SHRUTI network, based on the number of correct answers to questions.
Artificial development	The process through which generative and developmental systems are evolved.
Biological plausibility	The trait of a system in which characteristics of that system are computational analogs of biological processes.
Causal Hebbian learning	A form of *Hebbian learning* in which observation of one predicate instance before another suggests that the first predicate causes the second, and so connections between neural clusters representing the two predicates are strengthened.
Conflicting relations problem	A problem encountered when trying to learn two relations in which the antecedent is the same but the sign is different, e.g. $P \rightarrow Q$ and $\neg P \rightarrow R$. Strengthening one of these relations weakens the other. See section 3.1.2.
Conjunctive relation	A logical relation containing conjunction in the antecedent, e.g. $A \wedge B \rightarrow C$
Direct encoding	A genome encoding method in which the genome provides an explicit description of the phenotype's structure.
Distributed representation	A neural encoding in which one concept is represented by many neurons and one neuron participates in the representation of many concepts.
e-area	Area under the error-time curve during the development of a SHRUTI network.
Error	The difference between a SHRUTI network's maximum possible *accuracy* and the actual *accuracy* obtained.
Fully localist representation	See *Localist representation*.

General relational structure	A relational structure common across neural-symbolic reasoning models in which input neurons represent the antecedents, output neurons represent the consequents, and hidden nodes represent each relation. See section 2.3.3.
Generative and developmental system	A system that is represented using indirect encoding.
Group 0 network	A network that contains SHRUTI mediator structures and provides correct answers for all questions on conjunctive and non-conjunctive relations.
Group 1 network (fact structures)	A network that contains SHRUTI fact structures and provides correct answers for all questions on facts.
Group 1 network (relational structures)	A network that contains non-conjunctive SHRUTI relational structures and provides correct answers for all questions on non-conjunctive relations.
Group 2 network	A SHRUTI network in which connections only exist between enablers and collectors so that the answer provided for any given predicate is always the same (fig. 3.27).
Group 3 network	A network containing no connections so that all questions are answered [0,0] (unknown).
Group X genome	A genome that develops a group X network
Hebbian learning	A neurally plausible learning mechanism in which connections between neurons that fire together are strengthened [41].
Indirect encoding	A genome encoding method in which the genome provides a set of instructions for the gradual development of the phenotype.
Localist representation	A neural representation method is *fully localist* when each neuron encodes for only one concept and no others. However, a localist representation can be *partially distributed* if a neuron encodes for multiple concepts, but encodes for one more than it does for any others.
Long-Term Depression	A long-term decrease in synaptic strength resulting from a lack of stimulation.
Long-Term Potentiation	A long-term increase in synaptic strength following repeated synchronous stimulation.
Neural plausibility	The trait of a neural network in which characteristics of its neurons and connections are

	computational analogs of real biological neurons and their synapses.
Neural-symbolic network	A neural network that represents and processes symbolic information.
Neural-symbolic reasoner	A *neural-symbolic network* that performs reasoning tasks.
Non-conjunctive relation	A relation containing no conjunction in the antecedent, e.g. $A \rightarrow B$.
Number of updates	The total number of weight updates performed on connections in a SHRUTI network during the learning and development process.
Partially distributed	See *localist representation*.
Recruitment learning	A learning method in which neurons become recruited into a network by virtue of strong interconnections with other recruited neurons.
Recruitment threshold	The threshold which input signals to a neuron must cross in order to gain strength during recruitment learning.
Zero-error network	A SHRUTI network that correctly answers all questions it is presented with.

Genome labels for neurons

E_CON (Existing connection)	A neuron which shares an existing input or output connection with SELF (chapter 4).
E_INPUT (Existing input)	A neuron which shares an existing input connection with SELF (chapter 3).
P_CON (Potential connection)	A neuron which does not share an input or output connection with SELF (chapter 4).
P_INPUT (Potential input)	A neuron which does not share an existing input connection with SELF (chapter 3).
SELF	The neuron for which developmental actions are being considered by the genome.

Variables

α	The SHRUTI learning rate (equations 2.3 and 3.1 on pages 50 and 76).
hDec	Numerator for calculating α when decreasing weight through causal Hebbian Learning.
hInc	Numerator for calculating α when increasing weight through causal Hebbian Learning.
rInc	Numerator for calculating α when increasing weight through recruitment learning.

1. Introduction

Artificial Intelligence can be divided into two categories: *general* and *applied*, also known as *'strong'* and *'weak'* artificial intelligence respectively [87]. *General artificial intelligence* is concerned with the production of conscious thinking machines that match or exceed human-level intelligence, whilst *applied artificial intelligence* is more concerned with the development of problem-solving or reasoning programs that can be applied to real-life scenarios. This thesis is concerned with the former, i.e. approaches to general artificial intelligence in which models take the form of computational abstractions of structures and abilities of the human brain. If we wish to implement artificial thinking machines, perhaps we can begin by translating the processes involved in human cognition into computational architectures.

Two other sub-categories into which artificial intelligence can be divided are *symbolic* and *sub-symbolic* approaches. In symbolic systems, for example logic programs [43], concepts and operations on those concepts are associated with symbols so that it is relatively simple to interpret the current state of the system at each stage of execution. In sub-symbolic systems, for example artificial neural networks, the meaning associated with any individual component is not so intuitive, as the representation of any concept is often distributed across different components which may also participate in the representation of other concepts. Symbolic and sub-symbolic approaches are taken to both general and applied artificial intelligence.

Within the context of general artificial intelligence, logic programs (symbolic) and artificial neural networks (sub-symbolic) have both been used to produce models of human cognitive processes. Logic programs are used to represent reasoning and decision making processes by drawing conclusions from background knowledge in the form of rules and axioms. Artificial neural networks (ANNs) are abstractions of the network of biological neurons in the brain and can be used to map an input vector to an output vector, for example mapping an audio signal to a meaningful sentence in order to simulate human speech recognition [63]. ANNs are relatively noise resistant and adaptable to changing environments. Furthermore, by simulating biological neurons, ANNs exhibit a trait known as *'biological plausibility'*, a term often used to describe systems that capture some aspects of real biological systems. Although ANNs may be preferred for biological plausibility and adaptability when producing a cognitive model, one disadvantage of this choice of model is that the information they represent is difficult to interpret because it is distributed across the neurons in the network. On the contrary, logic programs may be used to model higher-level cognitive processes and are often preferred for interpretability. However they are generally not resistant to noise and are not so adaptable to changes in the environment,

although machine learning techniques such as Inductive Logic Programming [73] can be used to update a program's knowledge.

In summary, each approach has some advantages over the other and an ideal model would therefore combine symbolic and sub-symbolic approaches so as to combine the strengths of each and eliminate the weaknesses. Furthermore, human brains are neural systems that are capable of both symbolic manipulation and adaptation through learning, and therefore a neural network representation of symbolic systems would in fact be a more complete model of cognition. The field of *neural-symbolic integration* is concerned with such models [4, 21, 40]. This thesis is particularly concerned with *neural-symbolic reasoning* [21], in which ANNs are used to learn logical relations and reason on what is learned.

Although neuroscientific understanding of which brain regions participate in reasoning is growing [81], no conclusions have been made as to how reasoning is represented by the neurons themselves. Therefore any model of neural-symbolic reasoning can only be claimed to be biologically plausible to the extent that it captures some aspects of biological neurons and only serves as a *possible* model of how they might represent reasoning. Another representational issue is that of *localist* [11, 77] and *distributed* [80, 86] neural encodings. Localist models employ a one-to-one or one-to-many encoding in which a concept is represented by a fixed neuron or set of neurons which are committed to the representation of that concept alone. In a distributed model, the relation is many-to-many so that one concept may be represented by multiple neurons and one neuron may participate in the representation of multiple concepts. Although the distributed hypothesis is the most popular, neuroscientific evidence does not completely rule out localist representations and for the purpose of modelling cognitive behaviour, localist representations are often preferred [77]. The majority of neural-symbolic reasoning models are in fact localist and provide abstract neural representations of a range of cognitive reasoning abilities, including propositional [45] and predicate logic [34, 46, 95] and a range of non-classical logics including modal [20] and temporal [18] logics, among others [8, 16, 19, 21, 82].

Unlike traditional ANNs, which output real numbers, some models [48, 95, 102] explore neural-level representation even further by experimenting with representation using *spiking neurons* [64] that mimic the output of spiking action potentials emitted by biological neurons. *SHRUTI* [89, 95], a model of reasoning which employs localist representation, uses spiking neurons to form *variable-bindings* (bindings between predicate arguments and values that those arguments may take). The model demonstrates a biologically plausible variable-binding mechanism in which a predicate argument can be bound to a value by firing neuron clusters representing each in such a way that their spike trains are in synchrony with each other. In other words, if the spike train emitted by the cluster of neurons representing x from predicate $P(x, y)$ fires in phase with the cluster of neurons representing value a, x is said to be bound to a. SHRUTI propagates argument bindings throughout the network so that reasoning can be performed on predicate clauses. In addition to its use of spiking neurons, SHRUTI also exhibits biological plausibility through its use of *Hebbian learning* [41], argued to be the process through which associations are formed between biological neurons, and further biological plausibility is ascribed to the use of or-

ganised, repeated neural structures used to represent episodic facts, logical relations and other concepts [96, 120]. Organised structures in biological organisms are represented by genomes which represent those structures only once and reproduce them multiple times, perhaps with some variation [103, 116]. The developers of SHRUTI therefore argue that SHRUTI networks could also be represented in this way and that such a representation would complement the biological plausibility of the SHRUTI model [96, 120]. However, a genotypic representation of SHRUTI networks was not proposed.

To support this claim, this thesis presents the first such genotypic representation of the SHRUTI model, extending work originally presented by the author in [112]. The genotypic representation method described above is known as *indirect encoding*, in which the genome provides a set of instructions for the gradual development of the phenotype. The alternative representation method is known as *direct encoding*, in which the genome provides an explicit description of the phenotype's structure. However, indirect encoding is by far the most biologically plausible of the two, as biological DNA directs the growth of an organism through a set of prescriptive instructions [22]. Indirect encodings are used in the field of *generative and developmental systems (GDS)* to represent a range of models that may include repeated, similar substructures [13, 105]. In such situations, indirect encodings are preferred over direct encodings because each substructure need only be encoded once and expressed multiple times. This is not the case with direct encoding, in which each instance of the substructure would require its own encoding. In other words, indirect encodings are preferred for providing more compact representations. Furthermore, they are shown to provide *scalable* representations in that the size of the phenotype is independent of that of the genotype [56]. This is not true of direct encodings. Scenarios to which GDS have been applied include the generation of images [88], virtual creatures [9, 99] and neural networks [37, 38, 39, 47, 55, 106].

The biological plausibility of any neural-symbolic model, including SHRUTI, can be supported further by demonstrating that it can emerge from an evolutionary search, just as the brain developed through an evolutionary process. For this reason, this thesis explores the biological plausibility of SHRUTI networks further by exploring the evolvability of the SHRUTI genome. The field of GDS also involves the evolution of genomes that employ indirect encoding. The process of evolving generative and developmental systems is often referred to as *'Artificial Development'* [13], and this thesis presents the first investigation into the evolvability of SHRUTI networks using artificial development, extending work originally presented by the author in [113, 114].

In addition to supporting the biological plausibility of SHRUTI, two other motivations drive the evolution of SHRUTI networks. Firstly, it has already been suggested that artificial development could be used to discover neural models of general cognition [55, 71, 85, 106]; and indeed a complete model of cognition must include the ability to reason. Neural controllers for agents completing tasks in a range of problems in physical and virtual environments have been produced through artificial development [37, 38, 39, 55, 47, 106], but the artificial development of neural-symbolic networks has not been attempted to the best of the author's knowledge. Another motivation is that through evolution it may

be possible to discover new neural-symbolic models that contribute to the understanding of how reasoning ability is represented at the neural level. Current neuroscientific understanding of how reasoning is represented is at the higher anatomical level [81], and understanding of how the neurons themselves perform reasoning is inconclusive. With respect to both motivations, the findings in this thesis do not confirm or deny that new neural-symbolic models of reasoning can be found through evolution, because the research presented focusses specifically on the evolvability of the existing SHRUTI model. The problem of discovering new neural-symbolic models through evolution is the aim of future work. However this goal still motivates the work in this thesis because the feasibility of discovering these models through evolution can be understood further by exploring the evolvability of an existing model. Because a case has already been argued for genotypic representations of SHRUTI networks but not other neural-symbolic models, SHRUTI was preferred.

The main conclusions of this thesis are that SHRUTI structures can indeed be represented using a biologically plausible genome, that genomes for SHRUTI structures are evolvable to an extent, and that in future work, the use of distributed representations in the SHRUTI model would be worth exploring as a possible means of improving evolvability. The following paragraphs elaborate on these points.

The genomes used to develop SHRUTI networks were biologically plausible to the extent that they captured a number of aspects of real biological development. These include the use of gene regulation [116], cellular differentiation [103] and activity-dependent development [33] to direct the gradual development of phenotype networks over time through indirect encoding, just as DNA directs the process of neurogenesis [1, 12, 123]. The developed networks themselves exhibit the biologically plausible traits of spiking neurons [64], temporal coding [30, 60, 98], and Hebbian learning [41].

Initial evolutionary experiments which evolved genomes for constructing simple SHRUTI relations of the form $A \rightarrow B$ only evolved connections between neurons, but in later experiments that attempt to evolve more complex SHRUTI structures, evolved genomes could also produce new neurons. The results show that genomes for developing simple SHRUTI relations of the form $A \rightarrow B$ can indeed be produced through evolution, thus supporting the biological plausibility of these structures even further. However, although one of the goals of evolving these genomes was to discover alternative topologies to the SHRUTI model, no alternatives were found. These findings therefore show that SHRUTI must work in a very precise manner in order to perform inference correctly, and that the genetic instructions required to produce these networks are also very precise. The genomes discovered through evolution and those constructed manually to test the genome model were scalable in the sense that the same, fixed-size genome could be used to develop networks for logic programs of different sizes. This independence of the genotypic and phenotypic sizes is characteristic of indirect encoding. Furthermore, evolved genomes were capable of developing networks for logic programs other than those that the genomes were evolved to represent. The reason that these genomes were both scalable and generalisable is that no genome represents an overall network structure, but instead each genome encodes

a neural representation of a logical relation between two predicates which generalises to all logic programs. This repeated expression of substructure is also characteristic of indirect encodings. Because the genomes have been shown to be scalable and generalisable, it is argued that the genomes would be capable of developing larger logic programs than those which were used for the experiments described in this thesis.

Although the results show that simple SHRUTI relational structures can be discovered through evolution, further experiments showed that more complex structures were unsuitable for evolution. The structures in question are those used for representing episodic facts such as $Own(Mary, Book)$ (Mary owns the book), and conjunctive relations of the form $A \wedge B \rightarrow C$. The findings show that the evolutionary search space lacks the information necessary to identify individual substructures required for the assembly of more complex SHRUTI structures in later generations. In other words, these structures require multiple components to be in place before they show any improvement in behaviour, but individual discovery of these components is not reflected by a positive change in objective fitness. As well as showing that the evolvability of SHRUTI networks is limited, these results also emphasise the brittle nature of SHRUTI networks. SHRUTI networks behave in a very precise manner and each component is necessary for inference to be performed properly. This is supported further by the fact that alternatives to the SHRUTI topology were not discovered, as was the case when evolving the simple relational structures.

The findings suggest that because SHRUTI employs localist representation, a distributed alternative to SHRUTI might be more evolvable, for the following reasons. SHRUTI uses localist representation in which each neuron and connection is assigned a very specific and necessary role such that certain substructures cannot affect the output of the overall network (and therefore the objective fitness of the network) in any way until certain neurons or connections have developed. In a distributed representation, neurons and connections may share in the roles of others, and therefore it is more likely that the discovery of an individual structure can provide some (if only a small) influence to the output of the network without the support of others. This is not to say that distributed representations will definitely improve evolvability, only that it is a suitable avenue of exploration for doing so in future work. The difficulty in evolving a localist model like SHRUTI suggests that the discovery of other localist models of reasoning through artificial or natural evolution might also be unlikely. If distributed models are found to be more evolvable, this would support the biological plausibility of the distributed representation hypothesis. Testing the evolvability of distributed SHRUTI networks would be performed by attempting to evolve them using the same methods that were used to evolve the localist networks in this thesis, and then comparing the performances of the evolved distributed networks against those of the localist ones in order to observe whether any improvements have been made.

A preliminary investigation into the possibility of distributing representations in SHRUTI is included in this thesis and shows that it is difficult to distribute the representation of predicate arguments (e.g. x and y in the predicate $Q(x, y)$) across neurons in such a way that variable-bindings are not propagated ambiguously through the network. A variable-

binding is an assignment of values to predicate arguments, for example the bindings $x : a$ and $y : b$ assign the values a and b to x and y so that $Q(a, b)$ becomes a term in the logic program. Given a relation $P(x, y) \rightarrow Q(x, y)$, the truth of $Q(a, b)$ can be queried by propagating its variable bindings back to $P(x, y)$ and comparing $P(a, b)$ with what is known about $P(x, y)$. However, when these bindings were propagated to $P(x, y)$ according to a distributed neural representation of $P(x, y) \rightarrow Q(x, y)$, the bindings were sometimes propagated ambiguously in that it was not clear which value was bound to which argument of $P(x, y)$. For example, a was sometimes bound to both x and y in P, when it should only have been bound to x. The problem of ambiguous bindings should be overcome before any attempts to evolve distributed SHRUTI networks are made. Furthermore, if a solution to this problem is found, the binding mechanism used may extend to other neural-symbolic models of reasoning that share characteristics with the distributed SHRUTI model, in order to enable variable binding to be performed in these models also [34, 27, 28].

With respect to the future goal of discovering new neural-symbolic models of reasoning, the difficulties encountered in evolving SHRUTI suggest that the discovery of complex localist models in general is unlikely for the same reason, in that individual discoveries may not yield informative changes in objective fitness. If distributed representations do improve the evolvability of the SHRUTI model, then when attempting to evolve new neural-symbolic models, it would be worth giving the evolutionary search the freedom to explore models that exhibit some degree of distributed representation.

In the process of conducting the research in this thesis regarding the development and evolution of SHRUTI networks, a number of contributions to SHRUTI's existing Hebbian learning algorithm were also made. The original learning mechanism in [118] was intended to learn probabilities associated with relations. For example, a probability of 0.9 associated with the relation $A \rightarrow B$ is interpreted as $P(B|A) = 0.9$. Before attempting to develop and evolve networks for representing logic programs, an investigation into the limitations of the learning algorithm was carried out, as the original literature only demonstrates the algorithm's ability to learn small probabilistic relations that do not contain predicate arguments or negation [118]. The findings show that the Hebbian learning mechanism does extend to larger logic programs that contain predicate arguments, but struggles to learn relations when the target logic programs contain both positive and negative predicate instances. A solution is presented which enables the truth of all relations to be learned, but not always the probabilities associated with those relations. Nonetheless, the learning of truth alone was sufficient for evaluating SHRUTI networks produced through evolution and development, in order to support the claims made in this thesis. The reader is referred to [27, 28] for a working neural-symbolic model of probabilistic reasoning on first-order predicates. Furthermore, the original SHRUTI learning mechanism could not learn conjunctive relations of the form $A \wedge B \rightarrow C$, only non-conjunctive relations of the form $A \rightarrow B$. This thesis also presents the first extension of the learning algorithm to conjunctive relations.

The following sections summarise the aims, claims and contributions of this research, before providing an overview of the thesis as a whole.

1.1. Aims

1. To test the claim of SHRUTI's developers that owing to SHRUTI's use of repeated, similar sub-circuits, it is suited to representation by indirect encoding that enables the circuits to be pre-organised in such a way that enables its structures to be learned.

2. To argue that by using such a biologically plausible representation, a stronger case can be made for the biological plausibility of the SHRUTI model.

3. To investigate the biological plausibility of the SHRUTI model further by exploring the evolvability of SHRUTI networks under the proposed genome representation.

4. To argue a case for exploring the possibility of improving the evolvability and therefore also the biological plausibility of SHRUTI networks by moving from a localist representation to a more distributed one.

1.2. Claims

1. SHRUTI structures can be represented by indirect encodings and therefore the biological plausibility of SHRUTI networks is supported by an encoding method akin to that used by DNA.

2. Simple SHRUTI structures can be discovered through an evolutionary process and the discovered genomes that develop those structures are both scalable and adaptable to logic programs other than those they were trained on.

3. The discovery of more complex SHRUTI structures in an evolutionary search is highly unlikely because the fitness landscape lacks the information necessary for identifying individual components that improve fitness when combined in later generations.

4. The difficulty in identifying these components is a consequence of the fact that they perform very specific roles that other components depend on in order to affect the overall output. Because this is likely to be a consequence of the localist encoding used by SHRUTI, a move to distributed representation is a worthwhile avenue of exploration in attempting to improve the evolvability of the SHRUTI model.

1.3. Contributions

This thesis provides the following novel contributions to the fields of neural-symbolic integration and generative and developmental systems:

1. The first indirect encoding of SHRUTI genomes.

2. The first application of artificial development to the evolution of SHRUTI networks and neural-symbolic reasoning systems in general.

3. A detailed analysis of the SHRUTI fitness landscape and how properties of the SHRUTI architecture affect its evolvability.

4. A preliminary exploration of the ramifications of employing distributed representations in the SHRUTI architecture.

5. An investigation into the limitations of the original SHRUTI learning algorithm and improvements to this algorithm that enable the learning of logic programs containing negated predicates and conjunctive relations.

1.4. Thesis overview

Chapter 2 presents background material necessary to support the findings in this thesis. The review begins with an overview of biological principles of neuroscience and genetics on which models of neural-symbolic reasoning and GDS are based. The field of neural-symbolic integration is then introduced, providing a review of a number of neural-symbolic reasoning models but paying particular attention to SHRUTI. SHRUTI's use of repeated, similar substructures is also highlighted. The discussion of neural-symbolic integration is followed by an overview of GDS and artificial development, including a discussion on difficulties encountered when searching the space of developmental structures through evolution. In particular, genotypic discoveries necessary for improved fitness in later generations are not always reflected by positive changes in behaviour. The chapter concludes by arguing that the indirect encodings used by GDS are suited to representing SHRUTI networks because they can represent repeated structures in a biologically plausible way. This chapter includes discussions originally presented by the author in [115].

Chapter 3 explores the genotypic representation and evolvability of the simplest of SHRUTI structures: logical relations that do not include conjunction of antecedents and that can be learned through Hebbian learning. The chapter begins by reviewing the capabilities of the original SHRUTI learning algorithm in order to demonstrate that it is capable of learning the logic programs on which development and evolution were to be tested, and improves the algorithm in such a way that it can learn logic programs that contain both positive and negative predicate instances. The chapter then presents a genome model that uses indirect encoding to develop SHRUTI networks capable of correctly answering all questions on the non-conjunctive relations represented by the network. Furthermore, although the size of the genome is fixed, it is shown to be scalable to logic programs of different sizes. Discovery of these genomes through an evolutionary search using the NSGA-II algorithm [29] was also successful, thus supporting the biological plausibility of the SHRUTI model and its genome. Evolved genomes were found to be scalable and adaptable to logic programs other than those they were trained on because it is the repeated relational structure and not any particular logic network that is represented in an evolved genome. This chapter includes material originally presented by the author in [112, 113, 114].

Chapter 4 explores the genotypic representation and evolvability of SHRUTI structures that can be learned through a learning model known as *recruitment learning* [91, 93, 94]. These structures are *mediator* structures that enable conjunctive relations to be represented, and *fact* structures that enable the representation of episodic facts. Scalable genome representations were again successful in developing these structures so that they could correctly answer queries on background knowledge. However findings showed that these genomes were not suitable for evolution, unlike those presented in chapter 3, thus concluding that the evolvability of SHRUTI is limited. However analysis of these genomes and the fitness landscape were informative as to why this was the case so that future work can explore possible solutions. Mediator and fact genomes are considerably more complex than the non-conjunctive relational genomes presented in chapter 3, and the NSGA-II search could not discover them because the fitness landscape contained insufficient information to identify the necessary components or any proximity to them in genotype space. The individual components could not be identified because they cannot positively affect objective fitness when discovered individually, only when they are discovered together.

Difficulties in evolving SHRUTI are thought to be a consequence of the use of localist representation in which each neuron performs a specific role that may be unable to influence network alone. Distributed encodings may allow individual discoveries to make their own contributions to objective fitness and therefore exploration of a distributed SHRUTI network model would be worthwhile. Chapter 5 provides a discussion of the implications of a distributed SHRUTI model and presents preliminary results which demonstrate that distributed SHRUTI networks struggle to propagate variable bindings unambiguously. It is argued that a solution to this problem must be found before the evolution of distributed SHRUTI networks is attempted. The chapter concludes by suggesting future steps that should be taken in order to further investigate the idea of improving evolvability through the use of distributed representations.

Chapter 6 concludes the thesis by summarising the findings, discussing them in relation to the motivations and aims of this work, and proposing future work in addition to the exploration of distributed representations as discussed in chapter 5.

A dictionary of common terms used throughout the thesis may be found on page 15.

2. Background

The aim of the research presented in this thesis is to explore the evolvability of the neural-symbolic SHRUTI model using artificial development. This chapter presents an overview of relevant background material and extends a previous review also written by the author [115]. Because the aim is to produce a biologically plausible reasoning system in a biologically plausible way, the chapter begins with a review of biological preliminaries in section 2.1, including a discussion of what is meant by the term 'biologically plausible'. Section 2.2 provides a brief overview of a few psychological ideas that are also relevant to the work presented in later chapters. Section 2.3 introduces neural-symbolic integration, the goals of the field, and in particular introduces the SHRUTI model and why it might be particularly suitable for evolution using artificial development. Section 2.4 briefly introduces evolutionary computation before section 2.5 reviews generative and developmental systems and the process of artificial development through which they are evolved. Finally, section 2.6 concludes the background section by summarising key points and justifying the decision to attempt the evolution of SHRUTI networks.

2.1. Biological preliminaries

This section presents biological preliminaries required to support the ideas presented in this thesis and in the fields of neural-symbolic integration and generative and developmental systems. Sections 2.1.1 and 2.1.2 introduce the biological neuron and neural learning mechanisms respectively. Section 2.1.3 is particularly relevant to neural-symbolic integration because it provides an overview of neuroscientific findings with regards to how the brain performs reasoning. Section 2.1.4 provides a discussion on the issue of localist versus distributed neural representations and section 2.1.5 explains one theory as to how information is passed between neurons. Section 2.1.6 is particularly relevant to generative and developmental systems because it explains the developmental processes through which DNA produces living organisms. Section 2.1.7 concludes by defining what is meant by the term 'biological plausibility' in this work and explores the extent to which any computational model can be said to be biologically plausible.

2.1.1. The biological neuron

The biological neuron is a cell in the nervous system that processes and transmits electrical information [98]. The cell body is known as the *soma* and projects input and output connections known as *dendrites* and *axons* respectively. A neuron contains multiple dendrites

and one axon with multiple axon branches. Dendrite branches integrate input signals which are then processed by the soma. If the total input over a short period of time is great enough, the soma will fire a pulse known as an *action potential*, which is propagated to other neurons through the axon and its branches. The contact between the axon of one neuron (the *presynaptic* neuron) and the dendrite of another (the *postsynaptic* neuron) is known as a *synapse*. Synaptic transmission may either be electrical or chemical. The propagation of activity between neurons in the ways described means that activity at one neuron influences the activity at others, enabling the overall neural network to act as an information processing system. It is this very behaviour that inspired the *artificial neural network (ANN)* [42], described later in section 2.3.2.

2.1.2. Neural learning mechanisms

Two theories of neural learning mechanisms are relevant to the work in this thesis. In *Hebbian learning* [41], synapses connecting two neurons that frequently fire together within some time window are strengthened, thus forming an association between the two neurons. Synapses may also become strengthened through *Long-Term Potentiation (LTP)* [93], of which there are two types. In *homosynaptic* LTP, a synapse is strengthened when it receives a high frequency burst of spikes. In *associative* LTP, all synapses that synchronously participate in the activation of the postsynaptic neuron are strengthened. *Long-Term Depression (LTD)* is the opposite of LTP. In homosynaptic LTD, a synapse is weakened when it receives a low frequency burst of spikes and in *heterosynaptic* LTD, a synapse is weakened when it receives no presynaptic activity when other synapses of the same postsynaptic cell do. Because LTP is a rapid process, it is argued to be a suitable mechanism for learning and memory acquisition [84, 110]. Associative LTP is particularly relevant to this thesis because it forms the basis for *recruitment learning*, used by SHRUTI and related models to form episodic memories [91, 93, 94] and form associations between concepts to learn new types or predicates [96, 120].

2.1.3. Representation of reasoning in the brain

Neural-symbolic integration (section 2.3) is concerned with how biological neurons represent symbols in the brain. However current neuroscientific theory is focused on the higher anatomical level of the brain, and no conclusions have been made as to how the neurons themselves represent reasoning.

The hippocampus is a region of the brain which evidence strongly suggests is largely responsible for the encoding and retrieval of episodic memories [91]. Though not directly related to reasoning, episodic memory is relevant because a reasoning system, and SHRUTI in particular, may refer to episodic memories in order to draw conclusions. The SHRUTI literature makes the presumption that cortical circuits used in the reasoning process present queries to the hippocampal system and retrieve information from it [89, 95].

A recent quantitative meta-analysis compared findings of 28 neuroimaging studies of de-

ductive reasoning and identified cortical brain regions most commonly found to be associated with different types of reasoning [81]. Two theories of reasoning exist. *The Mental Model Theory (MMT)* suggests that deductive reasoning is performed by the manipulation of visuospatial representations in the brain regions associated with this purpose. *The Formal Rule Approach (FRA)* associates reasoning with those areas of the brain linked to the manipulation of language and syntax. The meta-analysis suggests that elements of both may be true. Sometimes visuospatial systems are engaged and at other times syntactic systems are engaged, depending on the type of reasoning concerned and other factors such as whether or not abstract concepts are used and the degree of certainty in reaching a conclusion. The study categorised reasoning into three types of argument:

- **Relational**, e,g: A is to the left of B and B is to the left of C. Therefore A is to the left of C.
- **Categorical**, e.g: All As are Bs, C is A, therefore C is B.
- **Propositional**, e.g: If A is true, then B is true. A is true, therefore B is true.

Relational arguments are associated with regions of the brain thought to be involved with spatial cognition, suggesting that relational reasoning is mapped onto spatial reasoning processes. Categorical arguments were found to activate areas of the brain associated with linguistic and syntactic processes. Finally, propositional arguments are linked with areas that could be associated with either visual or linguistic/syntactic areas of the brain. Although no consensus has been reached as to the extent of the involvement of both visual and syntactic processes, it is clear that different subsystems are involved with different types of reasoning. Furthermore, exactly how the neurons themselves represent reasoning can only be answered when the locations of neurons involved in this task have been confirmed.

It is also worth noting that human reasoning ability is limited, and therefore any system attempting to perfectly capture the representation of reasoning in the brain would be expected to operate under the same limitations. For example, humans can only perform a limited number of variable-bindings at any one time (7 ± 2 [95]), and therefore would struggle to perform reasoning in any scenario which requires more. Even within such limitations, humans still make mistakes. For example, a common mistake is *denial of the antecedent*, in which from the statement $P \rightarrow Q$, some may incorrectly infer $\neg P \rightarrow \neg Q$ [102]. To use a natural language example, $P \rightarrow Q$ could be interpreted as 'if it is raining then the grass will be wet'. If it is not raining ($\neg P$), it would be incorrect to assume that the grass is not wet ($\neg Q$) because the grass could be wet for any number of other reasons. For example, the grass may be wet because a sprinkler was left turned on.

2.1.4. Localist and distributed representation

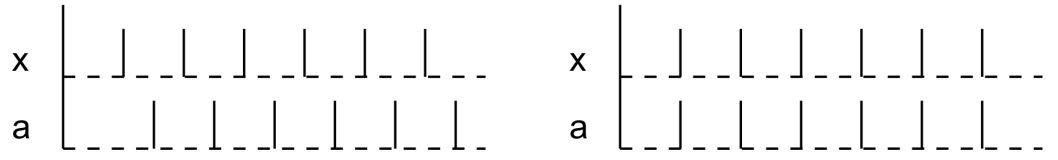
Another debate in neuroscience that pertains to neural representations in general is the question of whether neural representations are local [11, 77] or distributed [80, 86].

In a distributed model, the relationship between concepts and neurons is many-to-many, though various degrees of distribution have been proposed [80]. Dense distributed representations are the most extreme case, in which each neuron participates in the encoding of many different concepts. In coarse coding, a group of neurons encodes a range of related concepts. Finally, in sparse distributed coding, a neuron may encode a very small number of unrelated concepts. The distributed hypothesis does not limit all encoding in the brain to only one of these models, as different brain regions are likely to employ different degrees of distribution. In particular, representation in the hippocampus is thought to be sparse compared to other brain regions [67]. Sparse coding supports rapid learning and is therefore suitable for the learning of episodic memories in the hippocampus.

In a localist model, each neuron is associated with only one concept [11, 77]. In the most extreme case, often referred to as the *grandmother cell hypothesis*, neurons and concepts share a one-to-one relationship in that each concept is represented by only one neuron and is the only concept represented by that neuron. A common criticism of this representation is that it lacks graceful degradation in that the death of a neuron removes the corresponding concept from memory altogether. However the relationship in a localist model is not necessarily one-to-one, as it may be one-to-many in that any concept may be represented by a number of neurons that are only responsible for that concept. Another criticism is that individual neurons have been observed to respond to multiple concepts, but proponents of localist representation argue that localist models extend to ones in which a neuron participates in the encoding of multiple concepts as long as it is more responsive to one concept than it is to any others [77].

In general, a localist model is not localist by virtue of simply not being distributed. Localist models may exhibit some degree of distribution but what makes them localist is the idea that at some level of abstraction the model contains some neurons which participate in the representation of one concept more than they do in that of any other. It has been pointed out that many cognitive models employ localist representation and that many benefits of distributed representations such as generalisation, attractor behaviour and the ability to categorise are in fact possible using localist models if a suitable level of abstraction is used [77]. Furthermore, owing to the many-to-one correspondence between neurons and concepts, it is much easier to interpret the behaviour of a localist model than it is to interpret that of a distributed model. Even though localist models may not inform us as to exactly how information is represented at the neural level, they are sufficient to model and understand the behaviour of real cognitive systems in a way that can be interpreted.

In summary, even though the distributed hypothesis is a more widely supported theory of neural representation than the localist hypothesis [80], the latter has not been completely ruled out [11], and is in fact more suitable for cognitive modelling [77]. Considering that localism does not imply a complete lack of distribution and that varying degrees of distribution have been observed, perhaps localism and distribution lie at either end of a spectrum, and the true neural representation of reasoning lies somewhere on that spectrum. For clarity, the term *fully localist* will occasionally be used in this thesis to refer to particular localist representations that are not distributed to any extent, i.e.



(a) x is not bound to a (b) x is bound to a

Figure 2.1. Variable-binding by temporal synchrony of spiking neurons. When two node clusters representing variable x and value a fire in phase with each other, x is bound to a.

representations in which each and every neuron represents only one concept. The term *localist* is used more generally to refer to any model in which a neuron may represent multiple concepts but encodes more for one concept than it does for any other.

2.1.5. Temporal coding

Another issue concerning representation is the matter of how information is represented as it is passed between neurons. Although it is known that neurons communicate by propagating sequences of action potentials known as spike trains [30, 60, 98], there is no consensus as to how these spike trains encode information. One of the theories is *temporal coding* [30, 98], which is particularly relevant to the SHRUTI model. In temporal coding, information is encoded by the precise timing of action potentials, and it is suggested that correlation between spike trains of two neurons can be used to form associations between two concepts, for example variable-binding or pairing different features of a common object. Such *temporal synchrony* has been observed in the cat visual cortex [30]. Using variable-binding as an example, consider an variable x and a value a, both represented by different clusters of neurons. If the two clusters fire spike trains in different phases, then x is not bound to a (fig. 2.1a). However, if the two clusters do fire in the same phases, then x is bound to a (fig. 2.1b). The binding can therefore be interpreted as $x = a$.

2.1.6. Genetic development

Genetic development is controlled by instructions encoded in DNA [116]. Dawkins [22] explains that DNA is not *descriptive*, like a blueprint. Instead he explains that DNA is more like a recipe in that it provides a *prescriptive* set of instructions for the development of an organism's cells. How a particular cell develops depends on its current state and position in the organism, as governed by the process of *gene expression* in which proteins or other gene products are produced from information encoded in DNA. Different types of proteins may be produced when a gene is expressed: structural proteins that affect the structure of a cell, enzymes for catalysing chemical reactions, and transcription factors, which activate or inhibit other genes. In the latter case, expression of one gene can lead to a cascade of further gene expressions. The relationship between genes by which they are able to activate or inhibit one another is referred to as a *gene regulatory network*. However some genes may never be expressed at any point in time. Genes can be expressed are referred to as *exons*, whilst those that cannot are referred to as *introns*.

A cell's development depends on its state and location because the proteins that trigger the expression of its genes and therefore affect how it develops are received either from the environment or from neighbouring cells. This phenomenon is known as *cellular differentiation*, and leads to the development of different substructures in an organism at different locations in that organism, despite all cells containing the same DNA. Through cellular differentiation the DNA encodes the organisation of the organism as a whole, which includes the segregation of the brain into different regions [103].

The indirect nature of genetic encoding also enables any substructure encoded in the DNA to be expressed multiple times, perhaps with some variation due to cell differentiation. This means that similar substructures may be expressed multiple times despite being encoded only once in the DNA. Such repetition is clear from the existence of repeated structures in our body, for example teeth, fingers and limbs. Such repetition with variation even occurs internally in certain neural connectivity patterns, for example in cortical columns [103].

This thesis is particularly concerned with the development of neurons, known as *neurogenesis* [1, 12, 123]. Neurogenesis happens mostly during pre-natal development and neural development that takes place during the post-natal stage generally involves the formation of connections between existing neurons. However, there is recent evidence that new neurons are generated in the dentate gyrus region of the hippocampus and olfactory bulb during adult life [123]. Studies indicate that neural activity can influence development by regulating the expression of genes involved with formation and elimination of synapses and the branching of dendrites [33]. Therefore the environment and sensory experience can influence the development of neural connections.

2.1.7. Biological plausibility

Now that a number of biological mechanisms and processes have been reviewed, this subsection explains what is meant by the term 'biologically plausible' in this thesis. The human brain is argued to be the most complex system in existence [53]. The brain's neural network consists of approximately 10^{11} neurons all operating in parallel, each with approximately 7000 connections to other neurons. Furthermore, communication between neurons does not only involve electrical signals but also the propagation and processing of chemical neurotransmitters. No existing artificial neural network comes close to matching the complexity of such an organism, and using modern computational methods we could not hope to do so in the near future. Networks for representing logic programs in this thesis only required between approximately 30 and 2000 neurons, depending on the size of the logic program and the genome used to develop them. A precise modelling of human DNA and genetic development would be equally non-trivial, with approximately 20,500 genes in the human genome, resulting in combinatorial explosion in possible gene networks. The largest genome presented in this thesis contains only 74 genes (Fig. 4.27, page 157).

Given the complexity of actual biological systems and gaps in the understanding of how symbols and reasoning are represented by neurons, it is very difficult to argue that any

neural-symbolic model or generative and developmental system is biologically plausible in the sense that it is an exact model of its biological counterpart. All that we can hope to achieve are computational analogues that capture some aspects of the real biological systems and provide possible models of how such systems might behave. This is what is meant by the term 'biologically plausible' in this thesis. It has even been suggested that 'biologically *non-implausible*' may be a more suitable term[1]. The term 'neurally plausible' will also be used in a similar sense at various points throughout the following chapters, in reference to biological plausibility at the neural level.

By this definition of biological plausibility, the neural-symbolic networks developed and evolved in later chapters are biologically plausible in the sense that they capture the following aspects of biological neural networks: spiking neurons [60, 98], temporal synchrony [30], Hebbian learning [41] and Long-Term Potentiation [93]. The genomes presented in later chapters are biologically plausible in the sense that they capture the following aspects of biological genomes: developmental instructions [22], gene regulation [116], cellular differentiation [103], neurogenesis [1, 12, 123] and activity-dependent development [33].

2.2. Psychological preliminaries

In addition to biological explanations for the development of reasoning ability, some ideas from developmental psychology are also relevant because as an organism develops, its behaviour and capabilities change over time and the same is true of developmental neural networks. Piaget suggested that cognitive development occurs over four stages [78]. Of particular relevance to reasoning is the emergence of the ability to manipulate symbols in the second stage (ages 2 to 7) and the manipulation of variables in the fourth stage (11 onwards). Although Piaget's theories were very influential in directing further research into developmental psychology, they are quite old and a number of criticisms have since been made [100]. In particular, transition between stages is thought to be a lot more continuous and less discrete than Piaget proposed, and has also been observed to occur at different ages for different individuals depending on different factors such as environmental influence and memory capacity. The *information-processing theory* [69] supports a more continuous developmental model and suggests that improvements in cognitive ability are attributed to the development of physical resources in the brain such as the efficiency and capacity of neural structures. Furthermore, some types of thinking may be more advanced than others and the learning of different skills may even follow different trends. For example, some may follow a continuous increase and others may be step or stage-based, e.g. the progression from crawling to toddling and then to walking and running. Another trend is *U-shaped development*, in which certain abilities deteriorate before they improve, and has been observed in natural language acquisition [79].

[1] Though not cited in any papers, Murray Shanahan coined this term in a presentation at COGRIC, the slides for which can be found at: http://www.cogric.reading.ac.uk/presentations/murray_shanahan.ppt.

2.3. Neural-symbolic integration

Neural-symbolic integration concerns the representation and manipulation of symbolic information using artificial neural networks [4, 16, 21, 40]. Artificial intelligence is divided into two categories [87]. In *general artificial intelligence*, the goal is to produce intelligent, thinking machines that match or exceed human-level intelligence, and in *applied artificial intelligence*, the goal is to produce systems that perform problem-solving or reasoning tasks that may be applied to real-world scenarios. In both fields, there are generally two categories of solutions: *symbolic* and *sub-symbolic*. Symbolic systems such as logic programs [43] are often preferred for their interpretability, but in general are not robust to noise[2] and do not adapt well to unfamiliar data. For sub-symbolic systems such as neural networks [42], the opposite is true; sub-symbolic systems are relatively noise resistant and can adapt to new data, but the information encoded by those solutions is often difficult to interpret.

The example of artificial neural networks given above is a common approach to cognitive modelling. In artificial neural networks, individual nodes behave like biological neurons [42]. Information is distributed across a set of connection weights and is therefore difficult to interpret but adaptable through learning methods such as backpropagation [86] and Hebbian learning [41]. In cognitive modelling, the symbolic and sub-symbolic neural network approaches are often studied in isolation. However, humans are capable of both symbolic manipulation and adaptation through learning, so a more suitable model of cognition would be one in which elements of both symbolic and sub-symbolic approaches are present. The field of neural-symbolic integration presents such a unified approach, arguing that a true cognitive model exhibits the strengths and behaviours of both paradigms whilst eliminating the weaknesses, analogous to the interaction between the brain (sub-symbolic) and the mind (symbolic). Furthermore, the advantages of such unified systems show promise for real world application in applied artificial intelligence [8, 27, 28, 59]. The goals of the field of neural-symbolic integration can be summarised as follows:

1. To develop a greater understanding of how symbols are represented and manipulated in the brain at either the abstract or neural level by developing and testing biologically plausible neural-symbolic models.

2. To combine the strengths of symbolic systems and neural networks and eliminate the weaknesses of each in order to develop more powerful mechanisms for solving real-world problems.

3. To develop methods of extracting knowledge from trained neural networks so that information distributed across those networks can be understood.[3]

This thesis is concerned with the first of these goals, more specifically with regards to how the brain represents concepts and performs reasoning on them. The majority of this section will therefore focus on *neural-symbolic reasoning* systems that learn from experi-

[2] Although some would argue a case for probabilistic models [54, 83].
[3] A review of knowledge extraction methods may be found in [51].

ence using neural computation methods and produce interpretable, symbolic conclusions by reasoning on what is learned [21]. As argued above in section 2.1.7, no neural-symbolic reasoning system can be claimed to be completely biologically plausible. Questions of the anatomy of cognitive reasoning and the extent to which representations are distributed need to be answered before we can begin to understand how reasoning is represented by the neurons themselves. Any neural-symbolic model can only be claimed to be biologically plausible to the extent that it is a possible model of how reasoning might be represented, without claiming that it definitely is. All models discussed claim biological plausibility to some extent, whether that be by using neural networks to model the expressive reasoning capabilities of the brain [8, 16, 18, 19, 20, 21, 82] or by using more neurally plausible models that involve spiking action potentials and Hebbian learning, thus reflecting how biological neurons behave more closely [48, 89, 95, 102]. In either case, this thesis argues that the biological plausibility of any model is supported further by demonstrating that the model can be represented and produced through biologically plausible means: by evolving developmental genomes that use gene regulation, cellular differentiation, and other such biologically plausible traits listed in section 2.1.6 to instruct the development of the phenotype through indirect encoding. This thesis explores these ideas using SHRUTI networks [89, 95].

Before introducing particular neural-symbolic reasoning models, some preliminaries on logic programs and artificial neural networks are presented in sections 2.3.1 and 2.3.2 respectively. The majority of neural-symbolic reasoning systems employ fully localist representations which have some similarities in how relations are represented. These similarities are discussed in section 2.3.3. Each neural-symbolic reasoner makes different yet equally important contributions to cognitive modelling. The majority of work concerns connectionist models of different types of reasoning with the aim of eventually showing how these models could interact, thus working towards a unified model of cognitive reasoning. These are covered in section 2.3.4. This thesis is more concerned with the contribution made by the SHRUTI model, that of a neurally plausible means of variable-binding. The SHRUTI model is covered in section 2.3.5. Section 2.3.6 reviews some neural-symbolic reasoning systems that employ some degree of distributed representation. These are relevant because it will be argued later in chapters 4 and 5 that some difficulties encountered in attempting to evolve SHRUTI networks may be overcome by using more distributed representations.

2.3.1. Logic programs

A logic program [43] is a set of clauses of the form $L_1 \wedge \cdots \wedge L_n \rightarrow A$. Each L_i in a clause is a *literal*, where a literal is an *atom* or negated atom, and an atom is a formula that cannot be decomposed into smaller formulae. A is the *consequent* of the clause, and each L_i is an *antecedent*. In *first-order logic*, atoms are predicates that contain variables so that clauses such as $P(x, y) \wedge Q(x, y) \rightarrow R(x, y)$ may be formed. Any clause containing no variables is said to be *ground* and a logic program containing no variables is *propositional*.

Clauses in first-order logic may also be *quantified* to determine the extent to which a

Logic program	**Meaning**
$Give(x, y, z) \rightarrow Own(y, z)$	If x gives y to z, then y owns z
$Buy(y, z) \rightarrow Own(y, z)$	If y buys z, then y owns z
$Give(John, Mary, book)$	John gave Mary the book.
$Buy(Paul, z)$	Paul bought some z

Table 2.1. A logic program. For consistency with SHRUTI's notation with regards to quantification, z in the fact $Buy(Paul, z)$ is existentially quantified.

predicate is true for its variables. For example, the *universal* quantifier $\forall$ can be used to say $\forall x P(x)$ (all xs are P) and the *existential* quantifier $\exists$ can be used to say $\exists x P(x)$ (there exists at least one x that is P). However logic programming languages such as Prolog may exclude these symbols, assuming that all clauses are universal ($P(x) \rightarrow Q(x)$ is true for all x) unless a constant is used to represent the existential case ($P(a) \rightarrow Q(a)$ means that $P(x) \rightarrow Q(x)$ is only true when $x = a$). The SHRUTI literature, reviewed in section 2.3.5, always assumes universal quantification for clauses such as $P(x) \rightarrow Q(x)$, but contrary to standard notation a fact such as $P(x)$ is assumed to be existentially quantified if no constants are included to represent the existential case ($P(x)$ is true for some x).

An *interpretation* is an assignment of truth values to atoms, literals and clauses in a logic program. In first-order logic, this is accomplished by mapping symbols in the logic program to elements of a domain, thus assigning meaning to each symbol and the clauses they participate in. For example, if $P(x)$ is interpreted as $Fish(x)$ and $Q(x)$ as $Swims(x)$, then $P(x) \rightarrow Q(x)$ can be interpreted as $Fish(x) \rightarrow Swims(x)$ (all fish swim).

Notation may also be extended to other operators including those that enable the representation of non-classical logics. Although humans do display the ability to reason with non-classical logics, such logics are not relevant to the main body of work in this thesis because they are not represented in the SHRUTI model. However non-classical logics are briefly discussed later in section 2.3.4.

A logic program can be *queried* by asking it to confirm the truth of a predicate instance. For example, consider the logic program in table 2.1 and the query $Own(Mary, book)$ to be confirmed as true or false by the logic program. Based on the variable-bindings $y : Mary$ and $z : book$ and the clauses $Give(x, y, z) \rightarrow Own(y, z)$ and $Buy(y, z) \rightarrow Own(y, z)$, $Own(Mary, book)$ is true if $Give(x, Mary, book)$ or $Buy(Mary, book)$ are true. Because the logic program contains $Give(John, Mary, book)$, $Give(x, Mary, book)$ is true for some value of x and therefore by the relation $Give(x, y, z) \rightarrow Own(y, z)$, the logic program confirms that $Own(Mary, book)$ is also true. Queries are existentially quantified so that a query of the form $Own(Mary, z)$ asks 'Is there some z that Mary owns?', or in other words, 'Does Mary own anything?'. The response to this query would confirm that she does own something, and also identify what it is that she owns (the book) by returning $z = book$. If the program contained an additional fact $Own(Mary, car)$, the query would return both $z = book$ and $z = car$ to identify that Mary owns both the book and the car.

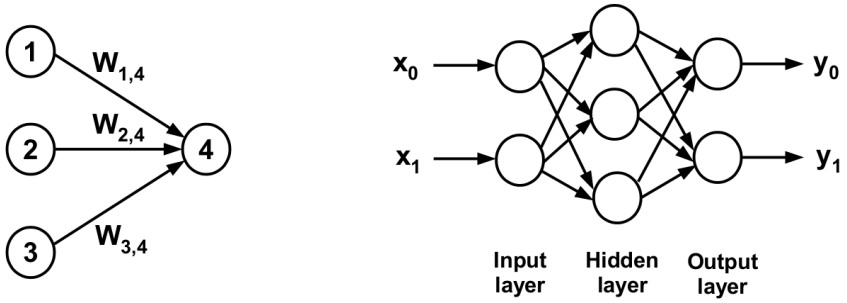
2.3.2. Artificial neural networks

Artificial neural networks (ANNs) consist of a set of connected nodes where each node is an abstraction of a biological neuron [42]. In traditional ANNs, a neuron may contain n inputs and m outputs. Connections between neurons are defined by a matrix W where element W_{ij} specifies the weight of the connection from neuron i to neuron j (figure 2.2a). Given a set of inputs I_i to neuron i, the input potential U_i can be calculated as the sum of weighted inputs so that $U_i = \sum_j W_{ij} I_{ij}$. An *activation function* takes the input potential U_i and determines the activation level A_i of neuron i. The activation function may be linear, non-linear or sigmoidal. In the case of non-linear activation functions, the activation is binary in that i activates only when the input potential is above a certain threshold θ_i.

ANNs are traditionally organised into an input layer which propagates activation to an output layer, often via a hidden middle layer (figure 2.2b). A network is *fully connected* when each neuron in one layer outputs to every neuron in the following layer. The network as a whole computes a function f on an input vector X and produces an output vector Y so that $Y = f(X)$.

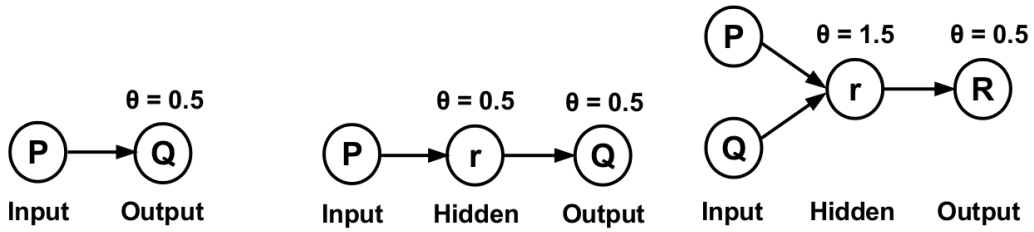
The weights of an ANN may be modified through a learning process. Supervised learning through *backpropagation* is particularly common, in which the error of the output vector Y is calculated based on a target vector and minimised by modifying W through gradient descent [86]. Neural networks may also be trained using Hebbian learning [41] in which connections between two neurons are strengthened when those neurons coactivate. This is argued to be a more biologically plausible alternative to backpropagation [77, 120].

Certain other traits of traditional ANNs are argued to lack biological plausibility. For example, there is no distinction between chemical and electrical signals and the propagation of real numbers across neurons is an abstraction of signal propagation in general. Furthermore, the propagated signals are not trains of action potentials or 'spikes' as they are in biological networks, although spiking neural networks do exist [64] and are used in the SHRUTI model [89, 95]. Nonetheless, what traditional ANNs lack in biological plausibility is not to be seen as a disadvantage because biologically plausible traits often lead to more complex models and traditional ANNs as they stand provide effective solutions to a wide variety of problems [76].



(a) A set of connection weights (b) A fully connected network

Figure 2.2. An artificial neural network



(a) $P \rightarrow Q$ without mediating rule node.

(b) $P \rightarrow Q$ with mediating rule node.

(c) $P \wedge Q \rightarrow R$ with mediating rule node.

Figure 2.3. The *general relational structure*. P, Q and R are atoms. Clauses in localist neural-symbolic reasoning systems include an input layer corresponding to antecedents, an output layer corresponding to consequents, and often a mediating hidden layer corresponding to the rule itself.

2.3.3. General relational structure

Neural-symbolic reasoning systems involve the use of ANNs to represent logic programs. The work is inspired by that of McCulloch and Pitts [68], who demonstrated the relationship between logic programs and ANNs by showing how logical conjunction, disjunction and negation could be represented using binary threshold units. Most fully localist neural-symbolic reasoning systems employ a similar structure in representing logical clauses, albeit used in different ways. Antecedent and consequent atoms in a clause are represented by separate nodes (neurons) in input and output layers respectively (figure 2.3). Clauses are implemented by connections from antecedent to consequent nodes, usually via hidden nodes corresponding to each clause. These mediator rule nodes enable conjunction by setting thresholds so that a fixed number of antecedent nodes must be active in order for the rule to fire (figure 2.3c). Different localist architectures employ this relational structure slightly differently as will be made clear in each of the relevant sections that follow, but in all cases, a relation takes the form of a two or three-layer structure as shown in figure 2.3. This common structure shall be referred to as the *general relational structure (GRS)* henceforth. The general relational structure is fully localist in the sense that each node maps to only one particular atom or clause.

2.3.4. Connectionist models

Most of the literature in neural-symbolic reasoning concerns connectionist representations of logic programs that adhere to the basic principles of traditional ANNs. These connectionist networks are used to implement the *neural-symbolic learning cycle* [4]. Background knowledge in the form of a logic program is translated into a connectionist network according to a translation algorithm. The network is then trained on further observations which may support information not contained in the original background knowledge. The knowledge encoded in the trained network can be extracted in symbolic form, thus producing an updated knowledge base that reflects not only the original background knowledge but also the information contained within the training data. Localist and not distributed representations are preferred for reasons highlighted in section 2.1.4, namely that localist models are just as capable of modelling cognitive behaviour as distributed models are, with the additional advantage of interpretability [21, 103]. Connectionist models exhibit

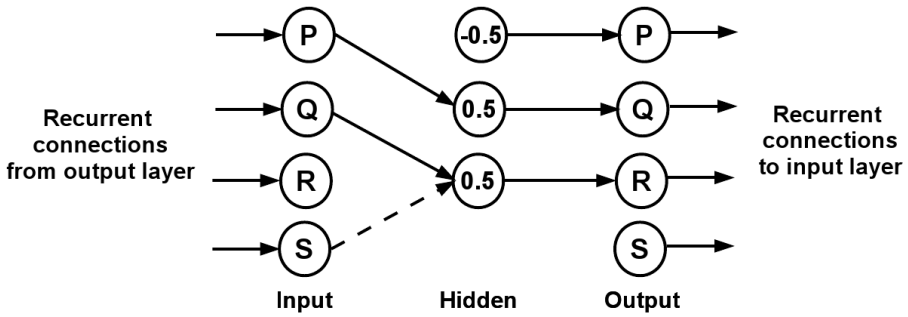


Figure 2.4. A core network representing the logic program P, $P \rightarrow Q$, $Q \wedge \neg S \rightarrow R$, assuming negation as failure. A negated antecedent in a rule is represented by a connection with negated weight, indicated by the dotted line. Diagram adapted from [5].

biological plausibility in the sense that networks of artificial neurons are used to model learning and reasoning across a range of domains in which the human brain is just as capable of doing so [8, 16, 18, 19, 20, 21, 82].

Propositional logic networks

Early connectionist networks known as *core networks* [45] use ANNs to encode clauses in propositional logic. Rather than providing an encoding of the logic program itself, Core networks attempt to capture the underlying meaning of the logic program. Core networks follow the basic pattern of the general relational structure (section 2.3.3) in that input, hidden and output neurons represent antecedents, relations and consequents respectively as shown in figure 2.4. The input and output layers are identical, containing a neuron for each atom in the logic program. Furthermore, the networks contain recurrent connections from each output neuron to the input neuron representing the same atom. These recurrent connections allow new atoms to become active at each cycle, and as activity continues to propagate over further cycles, the network state will converge on a least fixed point, providing that one exists. This least fixed point represents the truth of all clauses that can be derived from the background knowledge. Although the Core method enables convergence on the meaning of a logic program, no learning mechanism was provided to learn the original background knowledge or update it with new knowledge.

Non-classical logic networks

The Core Method and another connectionist model known as KBANN (Knowledge Based Artificial Neural Networks) [111] formed the basis for CILP (Connectionist Inductive Learning and Logic Programming) [16]. CILP's main contributions to connectionist neural-symbolic networks are the encoding of non-classical logics, the ability to train logic networks using traditional backpropagation, and the ability to extract knowledge from trained networks. Among the non-classical logics that CILP can represent are modal [20], temporal [18] and intuitionistic [19] logics. CILP encodes these logics by considering the possible worlds of the problem domain and representing each one in its own CILP network. Connections between each of these sub-networks enable transition between the

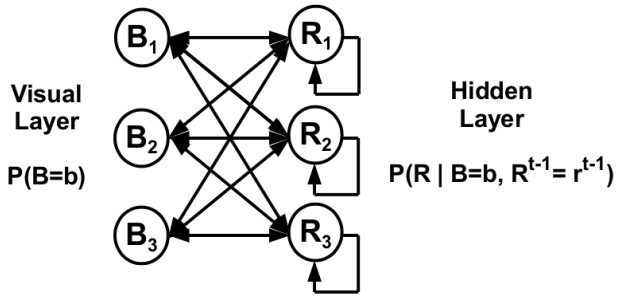


Figure 2.5. A Recurrent Temporal Boltzmann Machine (RTRBM) for calculating $P(R|B = b, R^{t-1} = r^{t-1})$, which is the conditional probability of a set of relations R between a set of observed beliefs b, given that set of beliefs and a set of previously hypothesised relations r^{t-1}. Diagram adapted from [27].

worlds. CILP has also been used to find solutions to abductive problems [82] and model normative reasoning, which has been applied to the control of robots in a RoboCup football game [8].

Probabilistic reasoning [54, 83], in which the truth of a relation has some degree of probability assigned to it, can be performed through neural-symbolic computation by using *Recurrent Temporal Restricted Boltzmann Machines (RTRBMs)* [27, 28], which have been applied to simulator training [28] and visual intelligence [27]. RTRBMs (fig. 2.5) are two-layer networks with recurrent connections in the hidden layer H, and are used to perform Bayesian inference. Nodes in the visual layer V represent a set of beliefs B and nodes in H represent hypotheses about a set of relations R. Each H_j calculates a posterior probability $P(R|B = b, R^{t-1} = r^{t-1})$ where b is a set of observed beliefs and r^{t-1} is a set of relations hypothesised in the previous time step. In other words, given a set of observed beliefs and the previous hypothesis, each H_j calculates the probability of a relation between those beliefs. It is the recurrent connections in the hidden layer that enable the previous hypothesis to influence the probability of the current hypothesis. The RTRBM samples rules r from $P(R|B = b, R^{t-1} = r^{t-1})$, and uses r to abduce a new set of beliefs by calculating the conditional probability of all beliefs ($P(B|R = r)$). The difference between observed and inferred beliefs can be used to train the network. When RTRBMs are used for visual intelligence in the Neural-Symbolic Cognitive Agent (NSCA) [27], beliefs take the form observed events and verbs that can be used to describe actions, and the hidden layer calculates the relationships between these.

Predicate logic networks

The Core Method (section 2.3.4) was extended to predicate logic [6, 46]. However the difficulty with representing predicate logic is that if a logic program contains function symbols, the program may contain an infinite number of ground atoms. Applying the original Core method would require an infinite number of neurons to represent all of these ground atoms. The proposed solution was to use an ANN which maps a real-number encoding of an interpretation of a logic program to another. An *embedding function* encodes the interpretation I into one or more real numbers and a hidden layer of sigmoidal units

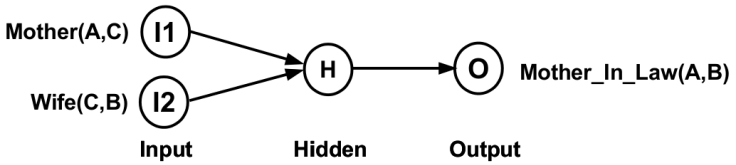


Figure 2.6. A first-order clause $Mother(A, C) \wedge Wife(C, B) \rightarrow Mother\_In\_Law(A, B)$ as represented by CILP++. Each first-order literal is represented by a separate node. Diagram adapted from

maps this onto another set of real numbers representing I', the immediate consequences of I. Through recurrent connections from output neurons to corresponding input neurons the networks iterates I until it settles into a state representing the meaning of the logic program. However, the solution may only be an approximation, owing to the inherent inexactness of real-number computation.

CILP has also been extended to predicate logic in a recent model called *CILP++* [34]. In a process called *propositionalisation*, a ground instance of each first-order clause is selected, which then acts as a propositional representation of that clause. This propositional representation is translated into a network structure using CILP's translation algorithm, and new first-order clauses can be learned from further examples using CILP's learning algorithm. Each literal in a first-order clause is represented by an individual node. Because certain roles in first-order clauses are shared by multiple literals, the representation of those roles is shared between the nodes representing each of those literals, and therefore CILP++ uses a form of distributed representation. For example, in the relation $Mother(A, C) \wedge Wife(C, B) \rightarrow Mother\_In\_Law(A, B)$ in fig. 2.6, although I1 and I2 represent different first-order literals, the concept of C is distributed across both nodes.

Although CILP++ provides a neural-symbolic means of representing and learning first-order clauses, a means of performing variable-binding in the neural architecture is not provided. Although each node *represents* a first-order literal, it treats that literal as a ground example of that literal. The network therefore behaves like the usual propositional CILP network, the difference being that a set of first-order clauses can be extracted from it. Variable-binding can then be performed on the extracted logic program, but variable-binding using the neurons themselves is a matter for future work. SHRUTI [89, 95], described below in section 2.3.5, provides a neural binding mechanism, and a possible means of adapting this to a distributed representation such as that used by CILP++ is discussed later in chapter 5.

Fibring neural networks

Connectionist non-classical reasoning systems allow for different types of reasoning to be performed using ANNs and a means of enabling these reasoning systems to interact in the form of an overall cognitive model has been proposed in the form of *fibred neural networks* [17]. Through so-called *fibring functions* one network may be treated as a neuron in another as shown in fig. 2.7. A *fibring function* is a function that translates a neuron or network's input or output. In other words, some fibring functions translate a neuron's

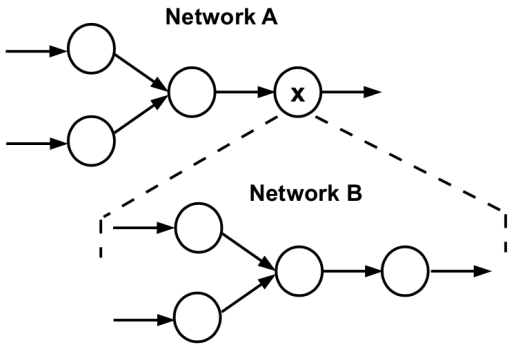


Figure 2.7. Fibring neural networks. The input to neuron x in network A is provided as the input to network B, the output of which is treated as the output of neuron x in network A. Diagram adapted from [17]

input into a form that a lower level network's input can understand and others translate a network's output into one that a neuron in a higher-level network can understand. By fibring networks, it is possible to perform recursive processing and increase the expressivity of connectionist networks to allow for the approximation of any polynomial function in an unbounded domain. This method provides another means of performing reasoning on first-order logic and also more generally provides a way of integrating networks that perform reasoning on different types of logic. Groups of networks integrated in this way are referred to as *fibred network ensembles* and point towards an overall cognitive model that explains how the brain is capable of performing different kinds of reasoning.

In summary, connectionist neural-symbolic networks have been proven to be a powerful neural-symbolic model as far as expressivity and application to real-world problems are concerned. Furthermore, the range of interacting capabilities offered by fibred network ensembles points towards a cognitive model capable of representing a range of reasoning abilities that humans are also capable of. Although connectionist models are biologically plausible in the sense that they model a range of human reasoning and learning abilities using an analogue of biological neural networks, one might argue that they are not as *neurally* plausible as some other models. As explained in section 2.1.7, neural plausibility is defined as biological plausibility at the neural level, whereas connectionist models exhibit biological plausibility at a more abstract level. Biological neurons encode information in spike trains, learn through Hebbian learning and not backpropagation, and the representation used is likely to be more distributed. Neurally plausible models such as those described later in sections 2.3.5 and 2.3.6 exhibit such traits. Although the connectionist models reviewed in this section do not exhibit these traits, they are still biologically motivated in that they are networks of neuron-like units that learn through induction, just as biological neural networks are capable of doing. They model a range of human reasoning and learning capabilities, and are therefore just as important for modelling cognition as neurally plausible models are. Furthermore, as argued at the beginning of this subsection on page 38, the level of abstraction that connectionist networks use is appropriate for modelling reasoning behaviour.

2.3.5. SHRUTI

SHRUTI[4] is a fully localist neural-symbolic reasoning model which is particularly concerned with modelling reflexive reasoning on causal relations between predicates in a neurally plausible way [89, 95]. The neural plausibility of SHRUTI is largely owed to the use of Hebbian learning [41] and temporal coding using spiking neurons [30, 98]. Some parallels between the anatomy of reasoning (section 2.1.3) and SHRUTI's structure are also shown in the literature. Furthermore, it is argued that SHRUTI draws even more biological plausibility from the fact that its structures are well suited to representation through a model of genetic development [96, 120]. The contribution of this thesis to the field of neural-symbolic integration is the first such genotypic representation of SHRUTI networks and an exploration of the evolvability of these genomes. Therefore to support the findings and conclusions presented in later chapters, it is necessary to provide a detailed description of the SHRUTI architecture here.

The use of spiking neurons and temporal coding allows SHRUTI to perform variable-binding using a technique referred to as *temporal synchrony*, in which a predicate argument can be bound to some entity by firing neural clusters representing each so that their spike trains are synchronous (fig. 2.1, page 31). These dynamic bindings can be propagated between clusters of nodes to encode relations between the concepts they represent.

SHRUTI can be configured for forward or backward reasoning. In forward reasoning, SHRUTI predicts the consequences of any given fact or observation. In backward reasoning, SHRUTI asserts the truth or falsity of predicate instances based on the antecedents of relations in which those predicates are consequents. In other words, backward reasoning in SHRUTI simulates the answering of a query in a logic program (section 2.3.1). This review only concerns how SHRUTI may be configured for backward reasoning, but a full explanation of SHRUTI's functionality may be found in the literature [89, 95].

SHRUTI architecture

SHRUTI employs a many-to-one localist relationship between neurons and concepts. Each node in figure 2.8 represents an ensemble of neurons which corresponds to a particular concept. For simplicity, SHRUTI can be implemented with one neuron per node, but at the expense of neural plausibility since even proponents of localist representation would argue that one-to-one representation in the brain is unlikely [11, 77]. In general SHRUTI uses a range of node types (table 2.2, page 45) and node functions (table 2.3, page 45). For the sake of clarity when explaining findings in later chapters, these node functions and types are assigned different labels as shown in the tables, though these labels are not necessarily used to describe these nodes in the SHRUTI literature.

Predicates are represented by clusters of nodes, i.e. super-clusters of ensembles of neurons.

[4] 'SHRUTI' is not an acronym but a Sanskrit word meaning "what is heard directly". The name was chosen because the SHRUTI developers draw a parallel between the dynamic pattern of acoustic energy required to encode "what is heard" and the dynamic patterns of neuron spikes which SHRUTI uses to propagate information.

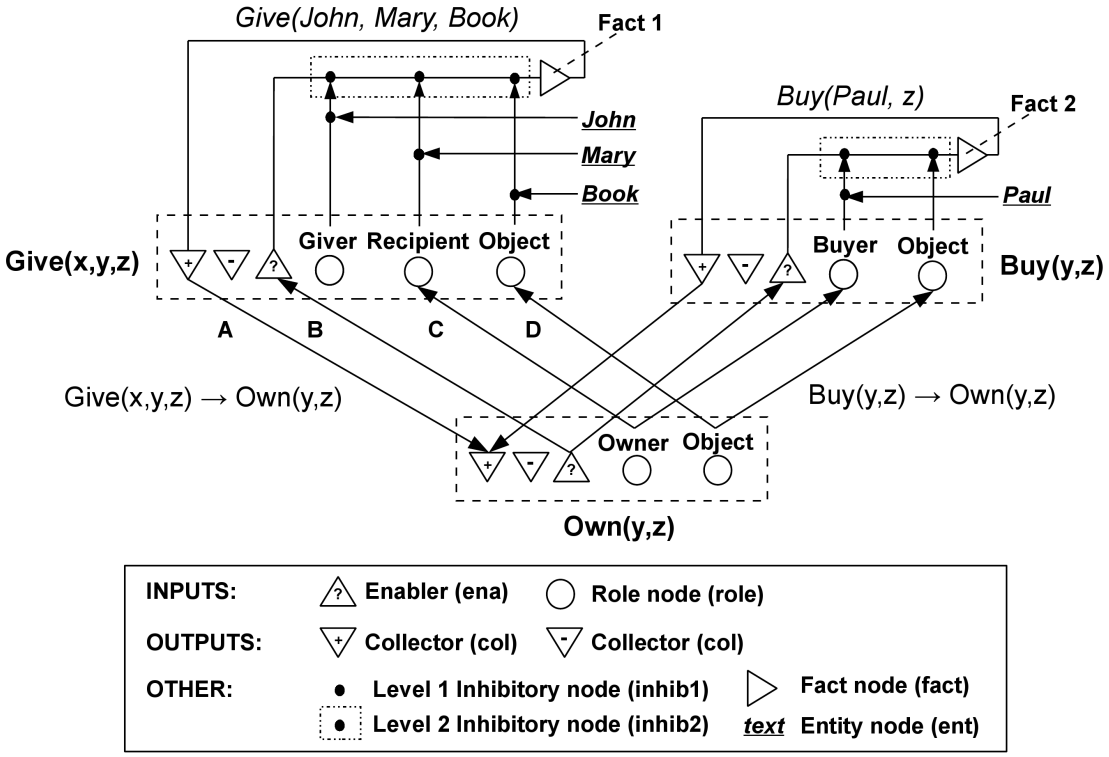


Figure 2.8. A SHRUTI network for two relations $Give(x, y, z) \rightarrow Own(y, z)$ and $Buy(y, z) \rightarrow Own(y, z)$. Inputs in the form of predicate instances to be queried can be provided at enabler and role nodes. Output at a positive or negative collector node corresponds to a predicate instance's truth or falsity, respectively. The propagation of activation is shown by the arrows. A detailed example of activation propagation during inference is given in fig. 2.10 on page 47. Diagram and example adapted from [95].

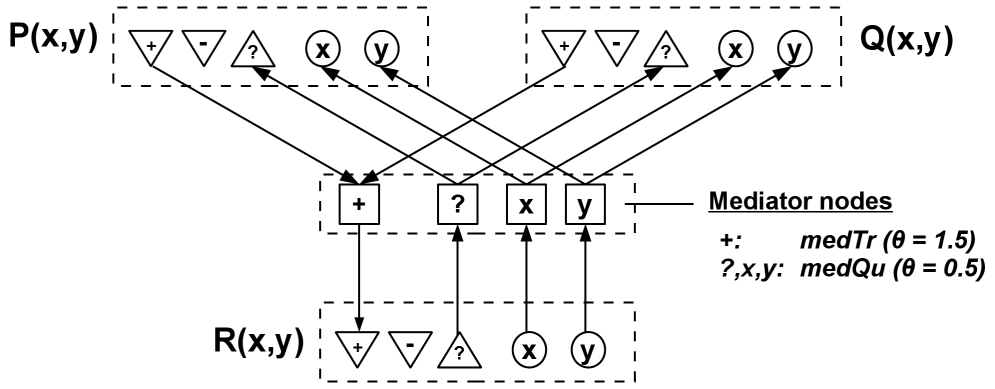


Figure 2.9. A SHRUTI network for the relation $P(x, y) \wedge Q(x, y) \rightarrow R(x, y)$, implemented using a mediator structure.

Predicate clusters contain one *role* node for every predicate argument. Each role node belongs to a family of nodes called m-ρ nodes (table 2.2). Each m-ρ node fires a spike train in the phase of its inputs, provided that those inputs are in synchrony with each other and that their sum is above threshold. They may also fire in multiple phases. A set of entity nodes (labelled '*ent*'), also of the family m-ρ, represent entities which may fulfil the roles of predicate arguments (e.g. *John*, *Mary*, *Paul* and *book* in figure 2.8). An entity is bound to an argument when the corresponding nodes are fired in synchrony with each other and a predicate instance takes the form of a set of role-entity bindings. For example, firing *Owner* and *Mary* in phase 1, and *Object* (from the *Own* predicate) and *Book* in phase

Label	Description
τ-and	Temporal AND node, fires an unbroken output signal (i.e. continuous and not phased) upon receiving sufficient above-threshold unbroken inputs.
m-ρ	Fires a spike train in phase with a set of above-threshold input spikes. May fire in multiple phases.
Inhibitory	Blocks signal propagation along a connection.

Table 2.2. Node types in SHRUTI

Label	Node function	Node type	Description
col	Collector	τ-and	Receives activation asserting the truth value associated with the current predicate instance.
ena	Enabler	τ-and	Receives activation to initiate a query; i.e. a search for the truth or falsity of the current predicate instance.
role	Role	m-ρ	Represents a predicate argument and receives activation in phase with an active entity node to form a dynamic binding between that predicate argument and that entity.
ent	Entity	m-ρ	Represents an entity that may be bound to a predicate argument.
medTr	Truth mediator	τ-and	Propagates truth from antecedent to consequent enablers.
medQu	Question mediator	τ-and or m-ρ	Propagates activations from consequent to antecedent enabler (τ-and) or collector (m-ρ) nodes as relations are searched for the answer to a query.
fact	Fact	τ-and	Fires when a predicate instance matches a fact in the background knowledge.
inhib1	Level 1 inhibitor	Inhibitory	Detects binding errors between role and entity nodes.
inhib2	Level 2 inhibitor	Inhibitory	Interrupts fact node activation if a binding error is detected in a level 1 inhibitor.

Table 2.3. Node functions in SHRUTI

2, creates an instance of $Own(Mary, book)$ (Mary owns the book). Predicate clusters also contain positive and negative *collector* nodes (*col*), which fire when the current predicate instance has been asserted as true or false respectively, and *enabler* nodes (*ena*), which fire when the truth of the current predicate instance is queried. Enablers and collectors belong to the *τ-and* (*Temporal-AND*) class of nodes, which do not fire phased spike trains but instead fire continuous, unbroken signals upon receiving continuous, unbroken signals from their inputs.

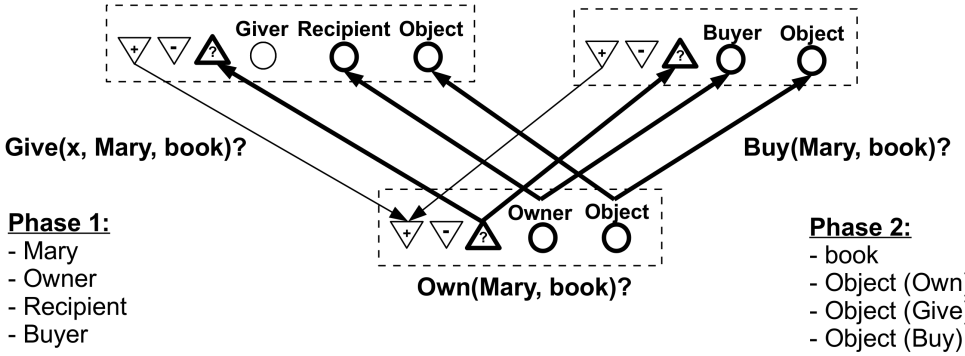
Relations between predicates are represented by connections between corresponding nodes, along which variable-bindings may propagate. For example, in fig. 2.8, the relation $Give(x, y, z) \rightarrow Own(y, z)$ is represented by connections between the collectors (A), between the enablers (B), and between role nodes that share bindings (C and D). Although existential and universal quantification in SHRUTI are both possible [90], relations are limited to universal quantification in the basic model described here. For example, $Buy(y, z) \rightarrow Own(y, z)$ is true for all y and z. Conjunction is possible by a mediating τ-and node which activates when it receives activation from the collectors of all of its antecedents and directs its own output to the collectors of all consequents (fig. 2.9). However, relations with repeated predicates in the antecedent (e.g.

$Father(x, y) \wedge Father(y, z) \rightarrow Grandfather(x, z)$) cannot be implemented without further circuitry that enables the multiple instantiation of predicates. This functionality is not covered in this thesis, but is described in detail in the literature [95]. Some SHRUTI implementations may employ entire mediator structures that propagate asserted truths (*medTr* nodes) and activation necessary for performing a query (*medQu* nodes) between antecedent and consequent predicate clusters. These mediator structures help to enforce type restrictions, which are also not covered in this review. Fig. 2.9 on page 44 shows a relation implemented with a mediator. Note that the basic structure of a relation adheres to the general relational structure described in section 2.3.3, in that activity propagates between antecedent and consequent nodes, possibly via hidden mediator nodes. The main difference is that the structure is repeated for each enabler, collector and role node instead of each antecedent, consequent and relation being represented by only one node.

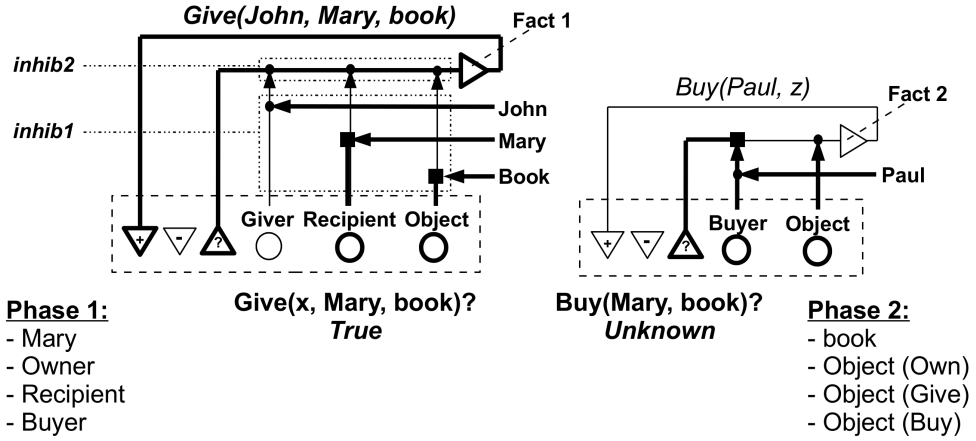
Long-term background knowledge is encoded by pattern-matching circuits that represent static facts. A *static binding* is a variable binding in long-term memory between an entity and predicate argument. For example, for the encoding of $Give(John, Mary, book)$ in fig. 2.8, the entity *John* is statically bound to the argument *Giver*. Static bindings persist through time, unlike *dynamic bindings*, which are temporary bindings created through the synchronous firing of neurons during inference. Facts in SHRUTI are existentially quantified so that, for example, the fact $Buy(Paul, x)$ is interpreted as 'Paul bought something'. A *fact* node only fires when the dynamic bindings of the corresponding predicate instance match a set of static bindings encoded by two arrays of inhibitory connections. For example, *Fact 1* in fig. 2.8 will only fire if $Give(x, y, z)$ is instantiated with any of the bindings from $Give(John, Mary, book)$ but no others. That is, *Fact 1* will fire for $Give(John, Mary, book)$, $Give(x, Mary, book)$, $Give(John, y, z)$, etc. but not $Give(Paul, Mary, z)$. Level 1 inhibitory nodes (*inhib1*) detect binding errors between role and entity nodes and propagate activation from the role nodes to the level 2 inhibitory nodes (*inhib2*) if bindings do not match. The level 2 inhibitors then block the activation of the fact node with this signal. The fact circuit used in the SHRUTI model is an abstraction that was later expanded upon in the form of the SMRITI model (System for the Memorization of Relational Instances from Transient Impulses) [91, 94], a model of episodic memory formation compatible with SHRUTI. Inhibitory nodes in SHRUTI correspond to more complex binding detector circuits in SMRITI that essentially perform the same function, inhibiting the propagation of the enabler's input to the fact node if any binding errors are detected. Otherwise, a successful set of bindings enables the fact circuit to propagate the truth of the relational instance to the predicate cluster and relational network in general.

To demonstrate how SHRUTI works as a backwards reasoner for answering queries, the example from section 2.3.1 will be used. The inference process is illustrated in fig. 2.10 on the following page and the node activation history is shown in fig. 2.11 on page 48. Consider the query 'Does Mary own the book?', which can be asked by creating an instance of $Own(Mary, book)$ as described above and firing the enabler of *Own* (fig. 2.10a). The bindings propagate along the role node connections of relations for which *Own* is a consequent so that *Mary* also becomes bound to *recipient* and *buyer*, and *book* be-

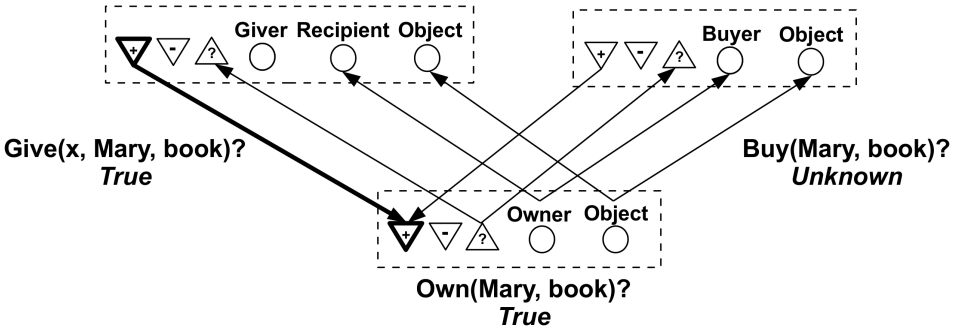
(a) The query 'Does Mary own the book?' is asked by firing the enabler (?) of Own and creating an instance of $Own(Mary, book)$ by binding $Mary$ to $Owner$ and by binding $book$ to $Object$ (Own).



(b) Bindings established in fig. 2.10a are propagated to $Give$ and Buy to present the queries 'Did somebody give Mary the book?' and 'Did Mary buy the book?'.



(c) Bindings for $Give$ and Buy are compared against the facts. Bindings match for $Give$ and its positive collector (+) activates to confirm $Give(x, Mary, book)$. Square inhibitory nodes indicate that activation has been inhibited.



(d) Activation propagates from the positive collector (+) of $Give$ to the positive collector (+) of Own, thus asserting that because $Give(x, Mary, book)$ is true for some value of x, so is $Own(Mary, book)$.

Figure 2.10. SHRUTI inference example. The complete circuit can be found in fig. 2.8.

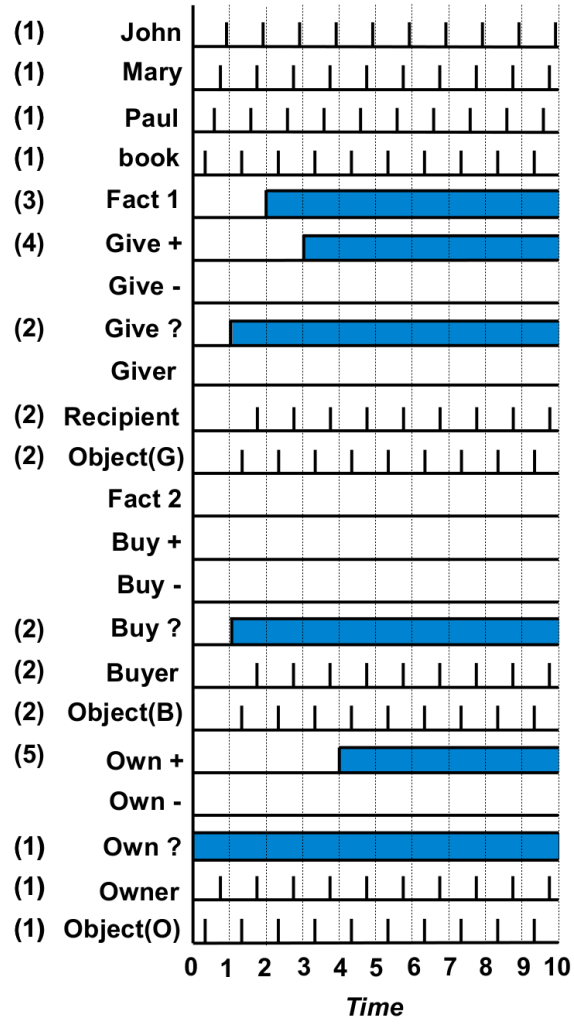


Figure 2.11. The flow of activation in the SHRUTI network when performing the inference process shown in fig. 2.10 to answer the query $Own(Mary, book)$? Numbers in brackets represent the order in which nodes fire. Mary, Owner, Recipient and Buyer all fire in one phase, and book and all object nodes fire in another. Enabler (?), collector (+, -) and fact nodes all fire in continuous, unbroken phase. Diagram and example adapted from [95].

comes bound to the *object* nodes of both relations (fig. 2.10b). Activity also propagates along the enabler connections to trigger a search for the truth of the antecedent instances. In other words, the network is now also asking 'Did somebody give Mary the book?' ($Give(x, Mary, book)$) and 'Did Mary buy the book?' ($Buy(Mary, book)$). In fig. 2.10c, the static bindings that represent the fact $Buy(Paul, z)$ (Paul bought something) are not matched by $Buy(Mary, book)$ and so output from the role nodes of Buy fire uninhibited by the inhib1 node associated with *Fact 2*. Activation from the role nodes therefore propagates to the inhib2 nodes, blocking the activation of *Fact 2* and thus indicating that the fact $Buy(Paul, z)$ neither confirms or denies the truth of $Buy(Mary, book)$. However, the static bindings of the fact $Give(John, Mary, book)$ (John gave Mary the book) do match the dynamic bindings at $Give$. Therefore the inhib1 nodes associated with *Fact 1* blocks the propagation of output from the role nodes of $Give$ so that *Fact 1* is free to activate. The activation of *Fact 1* propagates to the positive collector of $Give$ to assert that $Give(x, Mary, book)$ is true because $Give(John, Mary, book)$ is true (fig. 2.10c). Activation then propagates to the positive collector of Own and confirms that $Own(Mary, book)$ (Mary owns the book) is true because $Give(x, Mary, book)$ is true (fig. 2.10d).

At an anatomical level, the SHRUTI architecture has some biological plausibility. The collection of fact circuits corresponds to the hippocampus, where episodic memories are encoded and retrieved. These memories are sent to and retrieved from other cortical circuits that process them. However, as argued in section 2.1.3, exactly which cortical circuits participate in reasoning is unclear, though evidence suggests that circuits involved in both visual and linguistic tasks are also involved in reasoning [81]. In summary, the passing of activation between fact circuits and the reasoning system in SHRUTI is equivalent to the sharing of information between the hippocampus and other cortical areas.

The overall SHRUTI model has other capabilities such as type restrictions and the processing of queries other than 'yes or no' questions (for example, 'what did John give Mary?', i.e. $Give(John, Mary, x)$?), but these are excluded from this review as the artificial development of the necessary circuitry has not yet been attempted. Details of the more advanced capabilities of SHRUTI may be found in the literature [89, 90, 95, 117, 119].

Learning in SHRUTI

The SHRUTI literature also proposes means by which knowledge represented by SHRUTI networks can be learned [91, 96, 118, 120]. Of particular relevance to the work in this thesis are the learning of relations and facts.

Causal Hebbian learning (CHL), based on Hebbian learning [41], is used by SHRUTI for the learning of causal relations between predicates [118]. Furthermore, like the RTRBM [27, 28], CHL provides a neural-symbolic means of learning probabilities associated with logical relations, thus integrating the SHRUTI model with probabilistic reasoning [54, 83]. For example, connections in a SHRUTI structure for the relation $A(x, y) \rightarrow B(x, y)$ reflect $P(B(x, y)|A(x, y))$.

CHL works as follows. Consider again the relation $A(x, y) \rightarrow B(x, y)$. A sequence of event observations in the form of predicate instances are presented to the network. If an instance of a predicate A such as $A(a, b)$ is observed and then an instance of predicate B using the same bindings ($B(a, b)$) is observed to follow within a fixed time window, the weights of connections between their nodes are strengthened according to equation 2.1 in order to strengthen the relation $A(x, y) \rightarrow B(x, y)$ and reflect $P(B(x, y)|A(x, y))$, the likelihood that an instance of A is followed by an instance of B. However if B is not observed within the time window of A, the connections are weakened according to equation 2.2 in order to weaken the relation and reflect the likelihood that an instance of A is not followed by an instance of B. The learning rate α for any connection is inversely proportional to the number of weight updates that have been performed on that connection (equation 2.3) so that connection weights that have already been supported by a large amount of evidence are not easily changed. However, the number of updates must be initialised to a value greater than zero to avoid division by zero in equation 2.3.

$$w_{t+1} = w_t + \alpha(1 - w_t) \tag{2.1}$$

$$w_{t+1} = w_t - \alpha * w_t \tag{2.2}$$

$$\alpha = 1/\#Updates \tag{2.3}$$

If one does not wish to interpret probabilities and is only concerned with the binary truth of learned relations, as is the case in chapters 3 and 4 later on, true relations can be considered to be those for which connection weights are greater than or equal to 0.5, and false relations can be considered to be those for which weights are below 0.5. For example, if weights for $A(x, y) \rightarrow B(x, y)$ are above 0.5, then it considered to be true that when $A(x, y)$ occurs, $B(x, y)$ will follow. One observance of $(A(x, y), B(x, y))$ in the event sequence used for training is sufficient to raise the connection weights accordingly, but a later observance of counter-evidence ($A(x, y)$ occurring alone) can weaken the connection weights to below 0.5. The number of observances of $(A(x, y), B(x, y))$ required before the connection weights converge to a value above 0.5 depends on how often counter-evidence is also observed. More instances of $(A(x, y), B(x, y))$ must be observed than those of $A(x, y)$ occurring alone in order for weights to converge at a value above 0.5.

CHL was largely theoretical and only tested on very small propositional logic programs that did not include predicate arguments, as shown in figure 2.12. Furthermore, relational structures did not include mediator clusters and instead connected antecedent and consequent nodes directly. Before learning began, each node was connected to every other node of the same function (collectors to collectors and enablers to enablers) with weak initial weight values. A sequence of 500 events was generated according to the probabilities specified in figure 2.12a and presented to the network. After the network had observed all events, the connection weights loosely reflected the probabilities. For example, the weights of the connections from +:buy to +own and from ?:own to ?:buy reflect the probability of $Buy \rightarrow Own$ ($P(Own|Buy) = 0.75$) in that they are in roughly the same ball-park (0.67 and 0.81 respectively), but are not as accurate as they could possibly be. Then again, as argued earlier in section 2.1.3, humans are not perfect reasoners, and perhaps therefore a biologically plausible model of probabilistic reasoning should not be expected to be so accurate. Nonetheless, if one is interested in modelling probabilistic reasoning, and biological plausibility is not a concern, other models that are not neural-symbolic and are far more developed would be more suitable [54, 83]. Given the very limited environment in which CHL was tested, RTRBM [27, 28] shows more promise as a neural-symbolic model of probabilistic reasoning because it has been successfully applied to real-world scenar-

Relation	**Probability**
Buy	0.5
$Find$	0.5
$Buy \rightarrow Own$	0.75
$Find \rightarrow Own$	0.45

(a) Probabilities used to generate observation sequence

Source	**Target**	**Weight**
+:buy	+:own	0.67
?:own	?:buy	0.81
+:find	+:own	0.42
?:own	?:find	0.65

(b) Connection weights after Hebbian learning

Figure 2.12. Original test scenario for Hebbian learning in SHRUTI [118]. The probability associated with a relation can be interpreted as the probability that the consequent is observed after an occurrence of the antecedent, i.e. $P(consequent|antecedent)$, and learned connection weights loosely reflect this probability.

ios. However because this thesis is concerned with the artificial development of SHRUTI networks, it was necessary to use SHRUTI's CHL learning mechanism to learn relations. It was decided to test CHL on larger test scenarios than were used in [118] before attempting to integrate it with a model of development. These experiments are performed later in chapter 3. Furthermore, the experiments on trained networks in [118] only compare learned connection weights to target values and do not test performance. In other words, they do not test how well trained networks answer questions on the logic programs they represent. Trained networks are tested this way in chapter 3. Finally, CHL was not compatible with mediator structures and a solution is proposed in chapter 4.

Another learning method for SHRUTI, called *Utile Concept Learning (UCL)*, was proposed for learning new predicate or mediator clusters [120]. UCL involves a method known as *recruitment learning*, which has biological plausibility in that it involves the recruitment of nodes through long-term potentiation (section 2.1.2, [93]). Recruitment learning works as follows. A node is considered to be *recruited* when it shares strong connections with other recruited nodes in the network so that it can participate in the propagation of activation. A new predicate or mediator cluster is recruited (i.e. learned) when all of its nodes have been recruited through recruitment learning. In other words, a cluster is recruited by virtue of its strong connections to other recruited clusters. For example, coactivation of the predicate clusters for $Child(x)$ and $Male(x)$ could lead to the recruitment of another cluster which could be interpreted as $Boy(x)$. Coactivation of $P(x, y)$ and $Q(x, y)$ could lead to the recruitment of a mediator cluster that later enables relations of the form $P(x, y) \wedge Q(x, y) \rightarrow R(x, y)$ to be learned. In general, through Utile Concept Learning it is possible to learn a new concept by association of two older ones. However, UCL was only tested at an abstract level and a neural-level implementation was never proposed.

Although no learning mechanism exists for SHRUTI's fact structures, one was proposed for SMRITI [91]. SMRITI was produced to demonstrate how persistent episodic memories can be rapidly formed in a neural network anatomically similar to the hippocampal system. SMRITI is a very complex system, of which the fact circuits depicted in figure 2.8 only form an abstraction. Because this thesis is only concerned with this abstract model employed by SHRUTI, full detail of the recruitment of all of SMRITI's circuitry is excluded from this review. However a summarised explanation is included because learning in SMRITI is adapted to SHRUTI fact circuits later in chapter 4. Learning is performed through recruitment of nodes. The recruitment of circuits equivalent to level 1 inhibitor nodes (*inhib1*) is ultimately triggered by synchronous firing of the corresponding entity and role node inputs. The recruitment of circuits corresponding to the level 2 nodes (*inhib2*) and the fact node occurs when the output of the level 1 nodes coincides with output propagated from the enabler.

In general, SHRUTI has the capability to learn sets of facts, relations and other structures from a network of nodes with initially weak connections. However as the complexity of structures increases, the likelihood that those structures can be learned from a randomly interconnected network of neurons is reduced. SHRUTI's developers argue that some pre-

organisation of cognitive structures that enables those structures to be learned is required [96, 120]. Referring to the work of Marcus [65], the developers highlight that biologically plausible genetically-based processes such as generative and developmental systems (section 2.5) enable such pre-organisation, although a genetic model for the development of SHRUTI networks was not proposed. Such a model is the main contribution of the work presented in this thesis. Another reason for exploring such models and attempting to evolve them is that this thesis is concerned with working towards an understanding of how reasoning can be represented at the neural level. Biological neural networks are a product of evolution and therefore artificial evolution may be able to produce cognitive models that contribute towards such an understanding. Why SHRUTI is preferred over other models for this goal is elaborated further in section 2.3.7 below.

Although SHRUTI is preferred for its biological plausibility at the neural level, it employs local representations. The extent to which neural representations are local or distributed remains a matter of controversy (section 2.1.4). In either case, even proponents of localist representation argue that localist models may be distributed to some extent as long as any neuron encodes more for one concept than it does for any other [77]. The neural plausibility of SHRUTI could be supported further if it can be shown that its neurally plausible binding mechanism is also compatible with a more distributed representation. Furthermore, later findings in chapter 4 suggest that distributed representations may be a worthwhile avenue to explore in improving the evolvability of SHRUTI networks.

2.3.6. Distributed representations

In summary, distributed representations of neural-symbolic reasoning are of interest to this work for biological plausibility and for the possibility of improving the evolvability of SHRUTI networks. Findings in localist models make important contributions to the understanding of cognition and a move to distributed systems is not to say otherwise. While localist models are informative as to the capabilities of neural representations of symbols and how neural-symbolic cognitive modules could interact, distributed representations may be informative as to how knowledge is actually represented by the neurons themselves even if neural representations in biology are only partially distributed as some proponents of localist representation suggest [77]. In order to work towards a more generally distributed model, models in which both concepts and the relations between them are represented in a distributed way are of interest.

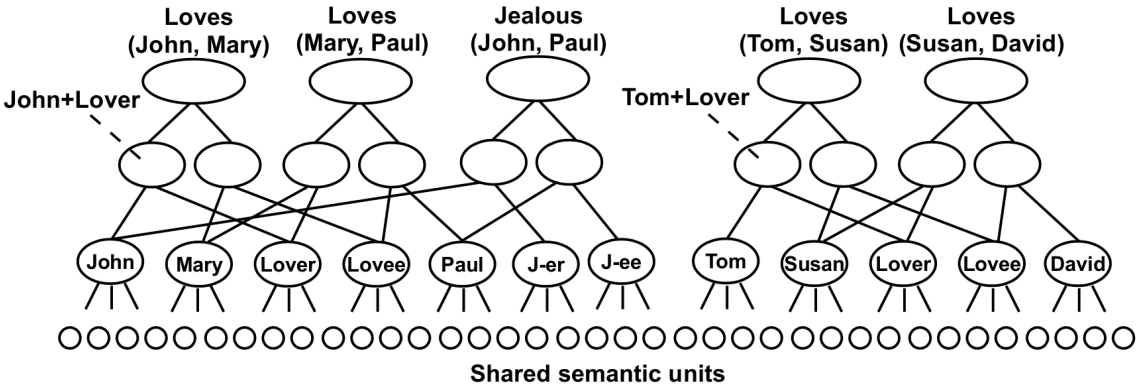
A number of neural-symbolic reasoners already include some form of distributed representation. Given that the primary aim of this thesis is to explore the evolvability of SHRUTI networks, the review of distributed models begins with other neural-symbolic reasoners that employ variable-binding by spiking neurons [48, 102]. However, these models only distribute the representations of the concepts which are reasoned upon, and the reasoning mechanisms themselves are not distributed. The review then considers models in which the reasoning mechanisms are distributed, even though these models do not use spiking neurons [3, 109]. Finally, other distributed neural-symbolic models for tasks other than

reasoning are briefly discussed with particular attention paid to similarities they share with neural-symbolic models that do perform reasoning [74, 75, 121].

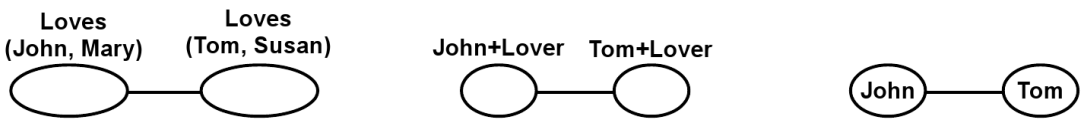
Spiking methods

LISA (Learning and Inference with Schemas and Analogies) represents concepts in a hierarchy in which concepts are constructed from lower-level units [48] (figure 2.13a). At the bottom of the hierarchy are a set of *semantic units* over which object representations can be distributed. For example, John is represented by bidirectional connections from the object unit 'John' to the semantic units 'male', 'adult' and 'human', and Mary is represented by 'female', 'adult' and 'human'. Because LISA distributes objects over shared semantic units, it is possible to perform reasoning by analogy. Nodes sharing semantic units will fire in the same phases and therefore Hebbian learning can be used to form mappings between those nodes so that they can be identified as analogous (figure 2.13b). Assume that one knows that because John loves Mary and Mary loves Paul, John is jealous of Paul. Assume also that one knows that Tom loves Susan and Susan loves David, but does not know that Tom is jealous of David. Nodes participating in *(loves(John, Mary), loves(Mary, Paul))* and *(loves(Tom, Susan), loves (Susan, David))* are analogous and so mappings are formed between corresponding nodes (figure 2.13b). Through these analogous mappings the network can infer that Tom is jealous of David (*jealous(Tom, David)*).

LISA is an example of a localist model that is partially distributed [77] because concept representations are only distributed in the sense that they are distributed over semantic units describing their features. The lowest level of abstraction is symbolic and the model does not degrade gracefully at this level. For example, removal of the symbolic unit



(a) Representation of the analogues 'John loves Mary, Mary loves Paul, John is jealous of Paul' and 'Tom loves Susan, Susan loves David'



(b) Examples of mappings formed between analogous nodes using Hebbian learning.

Figure 2.13. LISA - Learning and Inference with Schemas and Analogies. Diagrams and example adapted from [49].

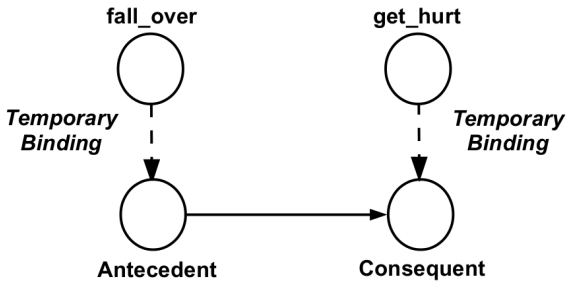


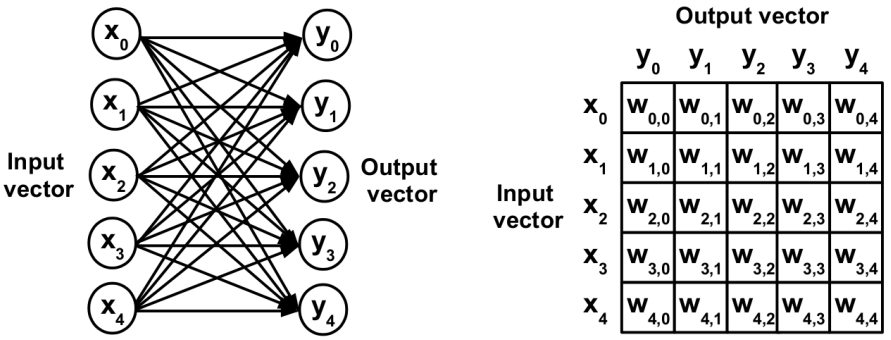
Figure 2.14. Inference in INFERNET [102]. Both antecedent and consequent are temporarily bound to concepts filling each role using Hebbian learning. Although the representations of *fall_over* and *get_hurt* may be distributed, the circuitry implementing the $A \rightarrow B$ rule to which those concepts are bound is local.

'human' still leaves concepts such as 'John' and 'Mary' intact, but eliminates the concept of 'human' entirely. Furthermore, even though concepts are in some sense distributed, the mappings between nodes that enable reasoning by analogy are local.

INFERNET uses sparsely distributed clusters to represent concepts [102]. The long-term knowledge base contains neural implementations of logic gates for encoding logical relations and answering queries. Working memory takes the form of temporary bindings between objects in the long-term memory so that reasoning processes encoded in the long-term memory can be used to perform inference on those temporary bindings. For example, given a general relation $A \rightarrow B$ in the long-term memory, a relation *fall_over* $\rightarrow$ *get_hurt* can be temporarily formed by binding A to *fall_over* and B to *get_hurt* through Hebbian learning (figure 2.14). However even though concept representations such as *fall_over* and *get_hurt* may be distributed, the nodes in the long-term memory to which these are bound, and other circuitry used to perform reasoning, are local. Furthermore, even though Hebbian learning is used to learn temporary bindings in short-term memory, long-term knowledge itself is encoded by fixed weights and no mechanism for learning long-term knowledge has been proposed. Learning is restricted to the working memory only. This is unlike SHRUTI, in which rules in the knowledge base can in fact be learned.

Associative Memories

Neither LISA nor INFERNET employ distributed representation in their encoding of relations, only in their representation of concepts. Systems based on *associative memories* (also known as *Lernmatrices*) [108] do not employ spiking neurons but do allow for distributed representation of relations as well as concepts. In an associative memory, symbols are represented as vectors for which each element can be represented by a neuron in the input layer and the output layer (figure 2.15). Neurons are fully interconnected as in a traditional neural network so that input vectors can be associated with output vectors. Associations between the elements of two different vectors are learned by using binary Hebbian learning to strengthen connections between neurons that coactivate. Given that a concept's representation is distributed over a set of vector elements, a set of element mappings therefore corresponds to an association between two concepts. Although they have more general applications, associative memories have been used to model reasoning.



(a) Network (b) Weight matrix

Figure 2.15. Associative memory (or *Lernmatrix*) for associating elements of an input vector with those of an output vector. The network can be represented as a matrix containing a weight $w_{i,j}$ for each input and output vector element combination. Diagram adapted from [121].

It is worth mentioning that RTRBMs [27, 28], mentioned earlier in section 2.3.4, are a form of associative memory.

Associative memories are a key component of the CLARION cognitive architecture [109]. CLARION explores the interaction between implicit (sub-symbolic) and explicit (symbolic) knowledge by representing the former using distributed methods, the latter using localist representation, and by allowing the two to interact. CLARION is a general cognitive architecture, and reasoning is among the cognitive abilities it is capable of representing, both at the explicit (top) and implicit (bottom) level. The bottom level contains identical sets of feature neurons in the input and output layers, similar to the feature sets used to distribute representations of objects in LISA. The set of weights between layers form an associative memory and encode relations between feature sets.

ARCA (Associative Rule Chaining Architecture) [3] is a rule chaining mechanism based on Correlation Matrix Memories (CMMs) [58], which combine correlation matrices and associative memories. Propositions and rules are represented as binary strings with neurons representing each element of the string. The input and output layers represent antecedent and consequent strings respectively. The middle layer corresponds to the set of rule strings. Through connections between layers, antecedents are associated with rules which are then associated with consequents.

Because input vectors are associated with output vectors, possibly via a hidden mediator layer, associative memories are effectively distributed equivalents to the general relational structure (section 2.3.3). Associative memories are even more closely related to connectionist networks (section 2.3.4) in that the output of an associative query can be recursively fed into the input of another in order to draw new conclusions at each iteration until all conclusions have been drawn.

It should also be noted that as with LISA [48], if each vector element is considered to represent an individual feature, loss of the corresponding node removes all knowledge of that feature. Therefore even though representations are distributed at some level, they will always be local at some deeper level. Nonetheless, associative memories still provide a move towards a more distributed model of reasoning in which both symbols and relations

are represented in a distributed way. While localist models are still relevant in cognitive modelling [77], distributed models such as associative memories may contribute to the understanding of how reasoning is represented at a less abstract level, i.e. how neurons themselves actually represent logical relations.

Other distributed Neural-Symbolic Methods.

The work reviewed so far only covers neural-symbolic reasoning systems. However work in the literature concerns distributed representation of other symbolic processes too. Some of the ideas discussed so far have similarities with other neural-symbolic cognitive models and therefore it may be that evolution of neural-symbolic reasoning systems supports the evolution of cognitive models in general or vice-versa. A more detailed discussion of these ideas and how they relate to artificial development may be found in [115].

A logical relation is a type of association between two or more concepts. The task of learning an association can be regarded as that of learning a new concept from old ones. This idea is supported in developmental psychology, with children learning new concepts through the equilibration of others [78]. SHRUTI's utile concept learning method works in this way, learning new predicates from older ones [120]. Aside from neural-symbolic reasoning but remaining within neural-symbolic integration, this learning of new concepts from associations between old ones has been applied to dimensionality reduction in design [75]. Dimensions are reduced by forming *chunks*, associations between design variables that have mutual effect in achieving a goal.

Associative memories have also been used in Neural-Symbolic Integration for visual intelligence [121]. Hebbian learning was used to learn associations between visual scenes as part of a production system [2]. Different states in the form of visual scenes are represented as binary feature vectors and associations between them are learned through Hebbian learning using input and target states. Using these associations, the production system is able to determine the necessary steps to take in order to reach a target state from the current state. The use of associative memories in both reasoning and visual intelligence is consistent with evidence that the same brain regions may be used for both tasks [81].

The Hebbian-based merge neural gas algorithm [66] has been used for learning action schemas that describe interactions between objects independently of schema labels [74]. For example, the network observes a dog chasing a cat and learns $A(x, y)$ instead of $Chase(Dog, Cat)$, distinguishing the two entities and the relation between them without learning what they are called.[5] This supports the cognitive theory that a mental schema does not immediately have a symbol associated with it [74, 75]. It is not a symbol that is learned but a generalisation of all the situations in which the concept that symbol represents has participated in, essentially forming an association between them.

With similarities such as learning by association and Hebbian learning, results from evolv-

[5]The Neural-Symbolic Cognitive Agent ([27], section 2.3.4), performs a similar task in that it also learns relations between visual events.

ing distributed reasoning systems may generalise to other cognitive models that involve learning by association. This is another reason to explore the evolvability of neural-symbolic networks.

2.3.7. Summary of neural-symbolic reasoning

All fully localist neural-symbolic reasoning systems employ similar structures in representing a logical relation, albeit in different ways. Antecedent and consequent propositions are represented by separate nodes, and connections between those nodes represent the relations between antecedents and consequents, sometimes via additional nodes corresponding to each rule. Different architectures implement this slightly differently. For example, Core networks represent antecedent, rule and consequent nodes in a single three-layer structure with recurrent connections from the output layer to the input layer (section 2.3.4). In SHRUTI on the other hand, each predicate and relation is represented by a cluster of nodes and a three or two-layer structure exists for each relation (section 2.3.5). In both cases, the two or three-layer general relational structure in figure 2.3 is applied.

Different neural-symbolic reasoners contribute to cognitive modelling in different ways. While CILP and fibring networks (section 2.3.4) contribute more to the expressivity of neural-symbolic reasoning by allowing for non-classical reasoning networks and the interaction between them, SHRUTI networks contribute more to neural plausibility by employing Hebbian learning and spiking neurons. This is not to say that either model is generally preferred over another, as they make different yet equally useful contributions towards cognitive modelling. An interesting direction for future research would be to explore how well the use of spiking neurons and temporal bindings translate to fibred non-classical logic networks in order to combine the advantages of expressivity and neural plausibility.

This thesis is more concerned with neural plausibility than it is with expressivity, aiming to evolve a neurally plausible model of reasoning through a biologically plausible means of evolution, with the potential for expanding the work to more expressive models and even cognitive models of other functions such as visual intelligence. Discovering neurally plausible models in this way may help provide understanding as to how biological neurons represent reasoning. Current knowledge as to how reasoning is represented in the brain is largely at the anatomical level [81]. Because SHRUTI networks exhibit some degree of neural plausibility, this thesis begins to explore the evolvability of neural-symbolic networks with these. Furthermore, the idea that all fully localist neural-symbolic reasoners share the same general relational structure suggests that conclusions made about the evolvability of SHRUTI may have implications for neural-symbolic reasoners in general.

However, the main reason for exploring the evolvability of SHRUTI networks in particular is that SHRUTI's developers have already argued a case for biologically plausible genetic representations of SHRUTI networks but this claim has yet to be tested [96, 120]. It was also for this reason that SHRUTI was preferred over LISA and INFERNET for exploring the evolvability of neurally plausible models. Genetic encodings allow for the reproduction of similar substructures and SHRUTI networks contain many repeated structures such as

relational and fact structures. A stronger case for the biological plausibility of SHRUTI networks could be made by demonstrating that they can be represented in a developmental genome and that such genomes can be discovered through an evolutionary search just as biological neural networks were.

Later findings in chapter 4 show that SHRUTI networks are difficult to discover through evolution, most probably because of the fact that they use localist representation, which is a reason to consider distributed alternatives to the SHRUTI model. Regardless of whether or not this would improve evolvability, compatibility with distributed representations would make an even stronger case for the biological plausibility of the SHRUTI model because this would demonstrate that SHRUTI's dynamic binding mechanism is compatible with both localist and distributed theories. Furthermore, even though the level of distribution in biological networks remains a controversial topic, even proponents of localist representation argue that localist models may be partially distributed [77]. SHRUTI's main contribution to neural-symbolic reasoning is the use of spiking neurons for forming dynamic bindings, so another interesting avenue for exploration would be how well this idea integrates with associative memories [3, 108, 109], which distribute the representation of both concepts and relations. This issue is discussed further in chapter 5.

2.4. Evolutionary algorithms

Before going into detail about evolution through artificial development, it is necessary to discuss evolutionary algorithms in general. Evolutionary algorithms exploit features of evolution and natural selection to discover optimal or near-optimal solutions to problems, particularly for scenarios in which exploration of every possible solution is expensive or intractable [52]. A random population of genomes is generated and each individual's suitability for the problem solution is measured according to a *fitness function*, where an improvement in fitness reflects maximisation or minimisation of the target objective, depending on the particular problem. Solutions are then selected at random for *variation*, a process through which individuals pass on their genes to the next generation. During the selection process, individuals are usually weighted by their fitnesses so that fitter solutions have a higher chance of passing on their genes. Common variation operators are *mutation* and *crossover*. In mutation, a parent is copied and randomly selected genes are changed to produce a new, child genome. In crossover, two parents are selected and copies of these parents randomly swap some of their genes to produce two children. The aim is that as this process of selection and variation repeats with each new generation of genomes, the average fitness of the population will improve until it converges at near-optimal fitness, suggesting that one or more near-optimal solutions have been found. The evolutionary process is stopped after a fixed number of generations or when fitness has converged.

However, selecting the best solution becomes more complicated when there are multiple objectives to be satisfied, especially if there is some trade-off between those objectives. In other words, there is no single solution that is optimal for all objectives. *Multi-objective evolutionary algorithms* [14] take all objectives into account and search for solutions that

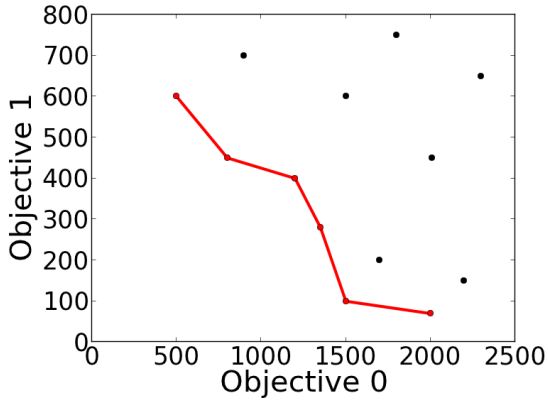


Figure 2.16. Solutions in a multi-objective minimisation problem. Points on the *Pareto front*, shown as a red line, are *non-dominated* in that they are not dominated by any other solutions.

are said to be *non-dominated*. If the goal is to maximise objectives, a solution i *dominates* another solution j if all objective values of i are greater than or equal to the corresponding objective values in j and at least one objective value of i is greater than the corresponding objective value in j. If the goal is to minimise objectives, the definition is the same except 'greater than' is replaced with 'less than'. With domination defined this way, a solution i is non-dominated if i is not dominated by any other solution. The *Pareto front* is the name given to the set of non-dominated or *Pareto optimal* solutions (figure 2.16), and the aim of the evolutionary search is to improve the objective values in this front at each generation. However solutions can no longer be selected for variation based on any individual fitness value. Instead, selection is often based on other measures. One such measure is *rank*, in which solutions are ranked based on their dominance of or by other solutions. Another method called *crowding distance* may be used as a tie-breaker when two competing solutions are equal in rank. In this case, solutions are selected based on their distance from other solutions in objective space in order to encourage a diverse set of solutions in the final Pareto front. After the desired number of generations or after convergence has occurred, it is the task of a human decision maker to choose their preferred solution from the Pareto front because no individual member of the Pareto front can be said to be better than any other.

The work in this thesis uses the multi-objective algorithm *Nondominated Sorting Genetic Algorithm II (NSGA-II)* [29] to evolve SHRUTI networks. In this algorithm, solutions are sorted into ranked non-dominated fronts where the Pareto front is the highest rank 0, and all members of any front dominate all members of lower-ranked fronts. At each generation, parent and offspring populations are combined, sorted by rank and then by crowding distance, favouring less crowded solutions. Edge points for each objective are assigned infinite crowding distance to ensure that they are always preserved. The best N solutions are selected to form a new population, where N is equal to the population size at each generation. This process of combining, sorting and selecting ensures *elitism*, the preservation of desirable solutions across all generations, which has been shown to improve the performance of genetic algorithms. From this new population, solutions are selected for variation by binary tournament selection, producing a new set of offspring so that the process may repeat. The full algorithm can be found in the literature [29].

2.5. Generative and developmental systems

The developers of SHRUTI argue that a model of genetic development would enable SHRUTI networks to develop in a biologically plausible way [96, 120]. Returning to Dawkins' example quoted in section 2.1.6, genetic development is not a blueprint of an organism's appearance but a recipe for its gradual production [22]. Genomes used in traditional evolutionary algorithms are closer to the blueprint model in that they are *descriptive* representations that explicitly encode the topology of the phenotype. This is known as *direct encoding* [97, 107]. *Generative and Developmental Systems (GDS)* [13, 105] are closer to the recipe model employed by DNA and are therefore more biologically plausible and relevant to the goal of this thesis. The genomes used by generative and developmental systems are *prescriptive* in that they provide a set of instructions for the gradual development of the phenotype. This is known as *indirect encoding*. In general, generative and developmental systems exploit many biologically plausible phenomena covered in section 2.1.6: gene expression, repeated substructures, cell differentiation and cell division. The field relating to generative and developmental systems and their evolution is sometimes known as *Artificial Development*, among other names [13][6]. For clarity henceforth, the term *generative and development systems* will be used to refer to systems represented by indirect encodings and the term *Artificial Development* will be used to refer to the process by which those systems are evolved.

Indirect encoding is a suitable means for representing and evolving SHRUTI networks because it is biologically plausible and because it enables structures to be encoded once but expressed multiple times just as DNA does. A SHRUTI network contains multiple instances of similar substructures, such as relational and fact circuits. In a direct encoding, each instance of each substructure would need to be encoded separately and as a consequence the size of the genotype would be proportional to that of the phenotype. Genomes using indirect encoding are *scalable* in that the size of the genotype is independent of that of the phenotype. Furthermore, when evolving direct encoding representations each instance of a substructure would have to be discovered separately in the evolutionary search, which is highly unlikely and very expensive. With an indirect encoding, each useful innovation only needs to be discovered once.

Structures defined through indirect encoding are often hierarchical in that some may form axes of development for others. Using the human body as an example, the spine forms an axis of development for the limbs, which form axes of development for the fingers and toes. Due to such hierarchical dependencies and the repetitive nature of structural development, small changes in the genome can result in large-scale changes to the phenotype. Because each level of the hierarchy is expressed in sequence, one change further up the hierarchy will affect structures further down. Even changes at the bottom of hierarchy have the potential to change the phenotype at a large scale because mutating one instance of a terminal structure mutates them all.

[6] Other names include: artificial embryology, artificial embryogeny, artificial ontogeny, computational embryology, computational development and morphogenesis [13].

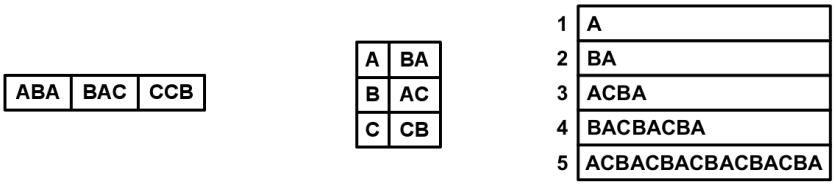
Generative and developmental systems have been applied to a range of scenarios, but the application of relevance to this thesis is that of evolving neural networks. It has been suggested that in addition to the development of solutions to control problems, neuroevolution may help us in the discovery of neural models of intelligence or general models of learning, especially since nature produces intelligent beings in this way [55, 71, 85, 106]. Neuroevolution is often used to produce neural controllers for agents performing specific tasks, but an ideal neural model in the long-term would be a general model capable of adapting to perform general tasks [71]. Neuroevolution enables researchers to understand the structure of a fully developed network based on its genome and environment. The same could be true of a general model of intelligence, which would include reasoning ability.

Developmental models of neuroevolution fall into two general categories: grammatical and cellular development models. Grammatical models are those that generate networks using some form of mathematical or symbolic language for recursively translating symbols into larger structures of symbols in the same language [39, 56, 85, 106]. Cellular development models take inspiration from the interaction of cells and gene products during biological development and therefore tend to be more biologically plausible [23, 24, 31, 32, 47, 57]. The genome model introduced in this thesis in chapters 3 and 4 borrows ideas from both approaches. In particular, the SHRUTI genome employs the abstract, tree-like structure of some grammatical models, and the diffusion of chemical neuromodulators as used by the cellular development models. Both approaches involve some form of gene expression and repetition of substructure, and these features are also included in the SHRUTI genome.

This section reviews ideas from both approaches to developmental neural networks. Section 2.5.1 reviews grammatical models and section 2.5.2 reviews cellular models. Section 2.5.3 concludes the review with a discussion of implications that the hierarchical nature of generative and developmental systems have regarding the ability of an evolutionary search to explore the fitness landscape.

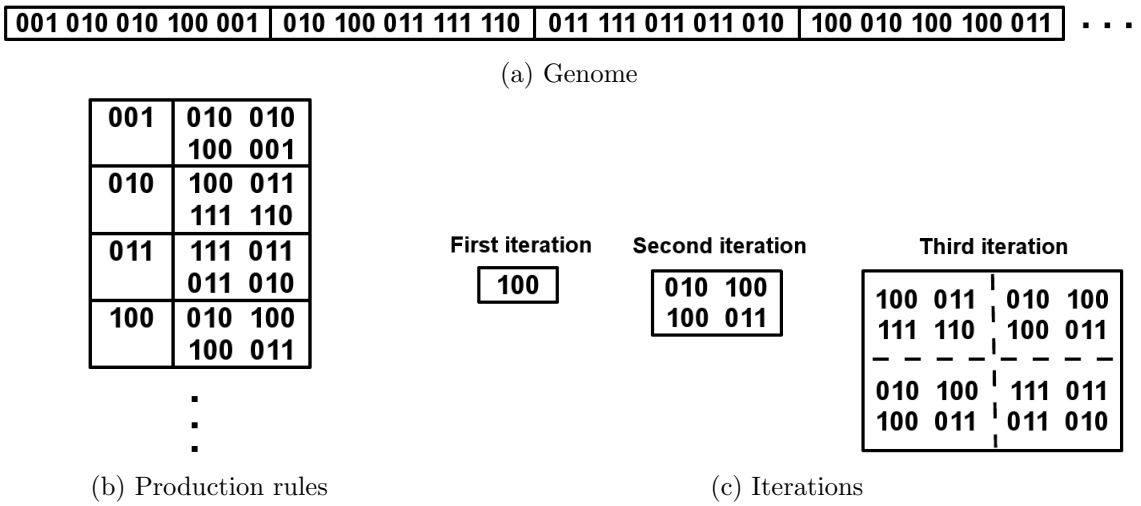
2.5.1. Grammatical encodings

Grammatical encodings are based on *Lindemayer systems (L-systems)* [62]. An L-system is a grammatical rewrite system defined as an alphabet of symbols V, an initial set of symbols ω, and a set of production rules P that map a member of V to a set of others. A string is produced by recursive application of the production rules to the symbols in V, starting from ω, over a desired number of iterations (figure 2.17).



(a) Genome (b) Production rules (c) Iterations

Figure 2.17. An example of an L-system represented as a genome. Beginning with the string 'A', new strings are generated iteratively according to the production rules encoded by the genome.



(a) Genome

(b) Production rules

(c) Iterations

Figure 2.18. Neuro-Genetic Learning (NGL) [56]. A connection matrix is developed from an initial string '100' according to the set of production rules. A function on the three-digit number at i, j determines whether or not neurons i and j are connected.

Kitano proposed a method of neuroevolution called Neuro-Genetic Learning (NGL) [56], in which grammar rules are used to develop connection matrices. The genome is composed of sets of five three-digit binary numbers (figure 2.18a). Each of these sets describes a production rule that translates the first number into a matrix of the other four (figure 2.18b). One set is defined as the initial seed from which progressively larger matrices are recursively developed (figure 2.18c). The final result is a connectivity matrix in which the value specified at i, j is used to determine whether or not neurons i and j share a connection. Kitano evolved these genomes and compared the results against the evolution of direct encodings for the same purpose. Networks produced with NGL outperformed those represented by direct encodings, especially for larger networks. Kitano argued that the superior performance of NGL was owed to the reproduction of useful sub-circuits and the compactness of the genome. Small genomes meant that the search space was smaller and therefore that convergence was faster with respect to the number of generations of evolution.[7] He also discovered that performance was greater when initial weight configurations were evolved rather than randomly assigned after evolution, as evolved weights would be closer to their optimal values, thus taking some of the workload from the learning algorithm.

Another grammatical encoding approach which found similar results was *Automatic Definition of Sub-neural Networks (ADSN)*, produced by Gruau [39]. Gruau used a grammar to describe cell-division processes rather than connection matrices. The grammar described a tree structure in which the root would describe the cell division of an initial single cell, and child nodes described that of the resulting child neurons. One approach was to define one tree for the development of the entire network. However Gruau also proposed ADSN, in which sub-trees were defined separately so that a leaf node of one tree could branch to the root of another, thus enabling repetition of sub-trees (figure 2.19). These genomes were evolved with the task of developing neural controllers for six-legged walking

[7] In terms of actual time, larger networks still took longer to evaluate than smaller ones. However, this was the case with direct encodings anyway.

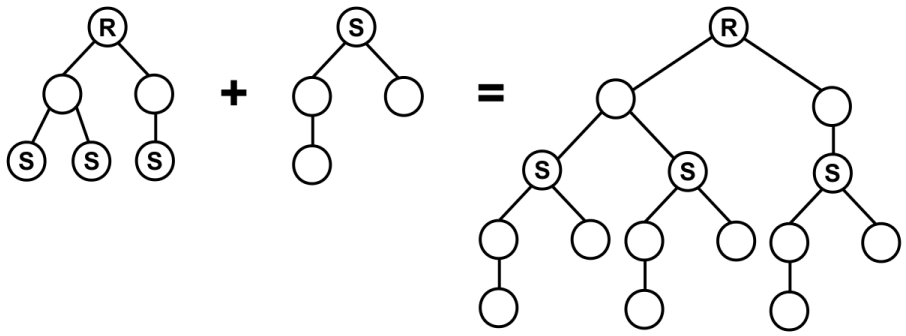


Figure 2.19. ADSN trees for describing cell division. The first tree describes the first set of divisions. Expression begins with the root node 'R' and when nodes labelled 'S' are expressed, they trigger cell divisions described in the second tree. Diagram adapted from [39].

robots for which distance travelled was the measure of fitness. Evolution was tested with and without ADSN. In other words, the evolution of tree structures for single networks was compared against the evolution of multiple, interacting sub-trees for interacting sub-networks. Once again, owing to smaller genomes and reduced search space, the solution employing repeatable substructures was found to yield the best performance. Gruau also tested the effects of evolving connection weights, using a stochastic hill-climbing algorithm to assign them. Evolution with hill-climbing was found to yield the better performance, also supporting the idea of evolving connection weights and training networks in parallel to development.

A more recent approach which follows a tree-like grammar that describes a hierarchy of operations is HyperNEAT [85, 106]. HyperNEAT genomes map x, y coordinate pairs to connection weights and HyperNEAT takes its name from the fact that the 2D connectivity patterns it represents are isomorphic to spatial patterns in hypercubes. HyperNEAT is actually a combination of two other models: CPPNs (Compositional Pattern Producing Networks) [104] and NEAT (NeuroEvolution of Augmenting Topologies) [107]. CPPNs are generative network structures that describe a hierarchy of spatial functions that map input coordinates to a level of intensity for the point at those coordinates (figure 2.20). Each node describes a function, for example symmetric or periodic. Due to the hierarchical nature of these functions, the output of one may provide an input to another, effectively combining the two. For example, one application of this method outside of neuroevolution has been image generation through function composition. Beginning with a symmetric function and feeding the result into a periodic one produces an image that is both symmetric

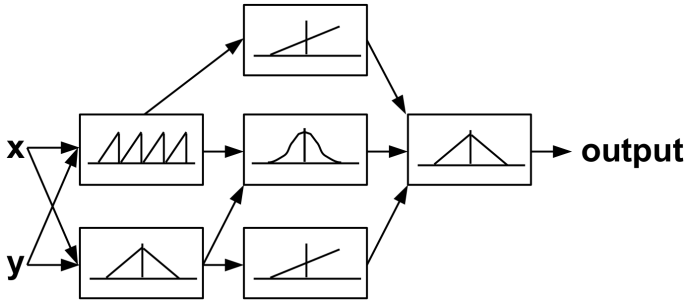


Figure 2.20. A Compositional Pattern Producing Network (CPPN) for producing composite functions that map a set of coordinates to some level of intensity. Diagram adapted from [104].

and periodic. Feeding a symmetric function into an asymmetric one produces imperfect symmetry. Performing an asymmetric function on a periodic one produces repetition with variation. Such hierarchical functional composition has been used to produce a wide variety of images using interactive evolution in which users manually select pictures for variation [88]. However, in HyperNEAT, the input received by a CPPN contains x, y coordinates of a pair of neurons in a substrate and outputs the strength of the weight between them. A weight below a fixed threshold is interpreted as no connection. A CPPN tree is therefore able to provide a generative, indirect encoding of a neural network. HyperNEAT uses its precursor, NEAT, to evolve these trees. Although HyperNEAT employs indirect encoding through its application of CPPNs, NEAT is a direct encoding method originally used to evolve neural networks directly. Because CPPNs represent tree structures, they are analogous to neural networks and therefore amenable to any algorithm capable of evolving them, such as NEAT.

In summary, HyperNEAT is yet another model that exploits the biologically plausible trait of repeated substructure for producing compact representations. CPPNs produce patterns in spatial mappings and the same is true when it is adapted to represent neural networks in that repeated substrates and connection patterns emerge just as they do in biological networks [103].

2.5.2. Cellular development models

Cellular development models are less abstract than grammatical encodings and take more inspiration from the behaviour of biological cells. However, the cost of this improvement to biological plausibility is the complexity of the methods used. In particular, connections between neurons are geometric and not topological. In grammatical encodings, connections either exist or do not, but in cellular development models, neurons have growing axons and dendrites. Cellular development models include those using gene regulatory networks [26] or Cartesian genetic programs [72].

As the name suggests, gene regulatory networks are based on gene regulation (section 2.1.6). Genes can influence the actions of other genes in the genome in addition to controlling the structure of the phenotype. For example, the expression of one gene may cause

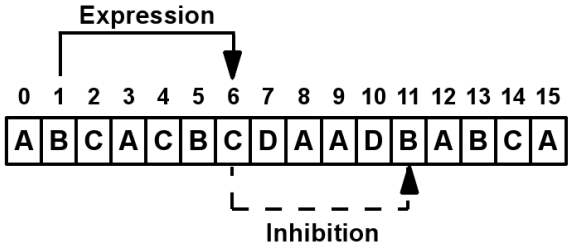


Figure 2.21. An example of gene regulation, in which the expression of one gene may cause another gene to be expressed or inhibited. Assume that each gene performs some action identified by the letter it represents. First, the expression of gene 1 triggers the execution of action B, followed by the expression of gene 6. The expression of gene 6 then triggers the execution of action C, followed by the inhibition of gene 11 so that gene 11 may not be expressed by any other gene.

another gene to be expressed or inhibited (figure 2.21). This may in turn result in the expression or inhibition of another gene, which leads to a cascade of other such actions. In some of these models, gene products which affect gene regulation and development may be passed between cells or received from the environment. Sufficient concentration of these gene products may lead to the production of further gene products, inhibition or expression of genes, or structural changes such as cell division or migration. Because these gene products are diffused throughout the structure, their concentration levels reveal positional information about cells in the system. This positional information may affect how those cells develop, as in the biological process of cell differentiation (section 2.1.6).

Eggenberger and de Garis [23, 32] both used cell differentiation to form shapes from cellular automata and showed that the method could be adapted for neural networks. de Garis used positional information in the neurons to direct the growth of axons between them. Individual neural modules were evolved this way and assembled manually with the aim of producing a model of the brain [24, 25].

In Eggenberger's gene regulatory networks, genomes were separated into structural and regulatory genes [31, 47]. Each structural gene was associated with a number of regulatory genes which would either express or inhibit them. The expression of regulatory genes depended on concentration levels of molecules called *transcription factors* passed between genes. Eggenberger's gene regulatory networks were used to develop a neural controller for a retina that focussed on incoming stimuli. The neurons differentiated so that neurons closer to the centre were more active in directing the retina towards stimuli [47].

Another approach which involves networks of genes and axonal growth is Cartesian Genetic Programming Computational Networks (CGPCN) [55], adapted from Cartesian Genetic Programming [72]. A Cartesian Genetic Program (CGP) is a directed graph represented by set of indices (figure 2.22). Each CGP contains the indices of its inputs, either external

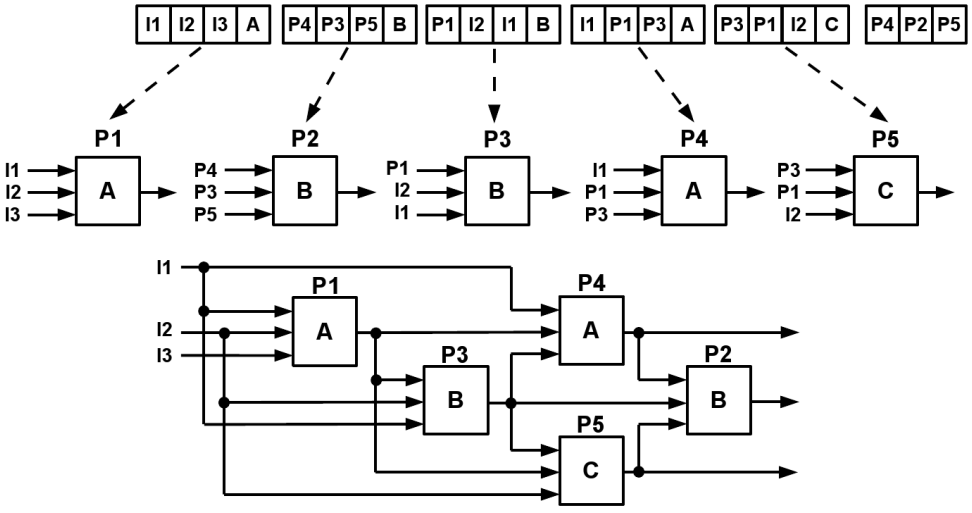


Figure 2.22. A network of Cartesian Genetic Programs (CGPs). The genome (top) lists a set of input indices and a function ID (A, B or C) for each CGP (middle) in a network (bottom). External inputs are represented by indices I1-I3 and all other indices correspond to inputs from CGPs. Indices of CGPs which provide external output are listed at the end of the genome. Diagram adapted from [55].

or from another CGP, and the index of a function it performs on its inputs. Extra indices at the end of the genome specify which CGPs provide the overall output for the overall network of CGPs. GGPCN uses seven CGPs, each with four genes. Three of these specify input indices and the final gene specifies which of four possible functions act on the inputs. All CGPs control the growth of the network in response to internal and external stimuli. The first three CGPs control signals to and from axons, dendrites and somas of neurons, modifying potential according to the specified function and then according to properties of health, weight and electrical resistance (inversely proportional to the length of the axon or dendrite). The fourth CGP controls network weights. The final three CGPs control the life cycles of the axons, dendrites and somas such as how and when they grow, shrink, die and reproduce. GGPCN was used to evolve controllers for agents playing games including Wumpus world, checkers and maze navigation. Future work aims to adapt the model to other applications with the aim of working towards a model of general learning [71].

2.5.3. GDS and objective fitness

Despite the advantages of biological plausibility and scalability offered by indirect encodings, recent findings demonstrate a particular difficulty in navigating the space of indirect encodings when using an evolutionary search directed by objective fitness. In summary, it is argued that objective fitness functions are highly deceptive with respect to this task [122]. Returning to the use of GDS to evolve images as an example, the image of a skull was produced through interactive evolution and serendipitous discoveries from human participants. However, attempts to reproduce the image using an objective fitness function that measured the difference between two images failed, even though the interactive evolution experiments demonstrate that CPPN grammar and variation operators must be capable of producing it. Using Picbreeder the skull was produced in 74 steps with a population of 15, but 30,000 generations were insufficient in a traditional evolutionary search using objective fitness. These results demonstrate that just because an evolutionary search fails to produce a particular result, doesn't necessarily mean that it is incapable of doing so, only that it is difficult. This point was also demonstrated using a technique called *novelty search* [61], which searches for novel solutions instead of searching for fit solutions. Where an objective-fitness based search struggled to discover a controller for a robot that could navigate a particular maze, a search for novel solutions was successful in doing so.

The problem in evolving both the skull and the robot was argued not to be the fitness function, but the necessary order of functions that produce the target solution. For ex-

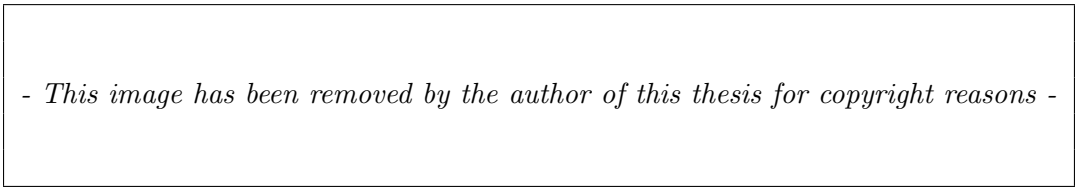


Figure 2.23. Necessary stepping stones towards discovering the image of a skull using Picbreeder, which uses CPPNs [104] to generate the level of intensity for a point in the image. Earlier stepping stones show little resemblance to the final image. Images taken from [122]

ample, the skull image began life as a simple symmetric function on which further detail of the skull was defined. This symmetry higher in the hierarchy was an essential requirement in order for the skull to be produced. However, the symmetry alone, and many of the further stepping stones towards the skull, did not resemble a skull and therefore the objective comparison operator could not recognise the stepping stones as precursors to the target image (figure 2.23). The same results were observed for many other images.

The claim is that the very action of setting images as objectives is what caused them to fail to be reproduced. Changes essential for further progress from any one point in genotypic space may not yield the desired improvement in objective fitness. In other words, movement in the correct direction in genotypic space does not necessarily imply movement in the correct direction in objective space and vice-versa. Returning to the skull example in fig. 2.23, each image from left to right is produced by a different and necessary step in genotypic space, each one genotypically closer to the genotype that produces the image of the skull. However, the difference (as measured by a difference operator) between the target image and the image at each step does not necessarily decrease as each step is taken, and therefore objective fitness (measured as the difference) does not move towards 0. The objective-based evolutionary search had the aim of minimising objective fitness and therefore favoured discoveries which reduced objective fitness. Because the genotypic steps necessary for producing the skull did not reduce fitness, they were not identified as useful discoveries by the evolutionary search.

One might argue that the issues of misleading stepping stones is essentially that of becoming stuck in local minima in direct encodings, which traditional evolutionary computation tackles through operators such as crossover. However the argument is not that these issues do not occur in direct encodings, but that they are a greater problem for indirect encodings because owing to the hierarchical and repetitive nature of the genomes, the difference in objective behaviour of adjacent stepping stones can be significantly large. Furthermore, in searching for the target phenotype, the minima should not be avoided because the minima are necessary for the development of the target phenotype, just as symmetry is a necessary stepping stone towards discovering the image of the skull. Novelty search [61], discussed above, is argued to be a means for discovering desirable solutions without constraining the search by objective fitness. Novelty search has been shown to discover solutions that searches using objective fitness function were unlikely to find. Furthermore, because such a diverse range of solutions are explored, desirable solutions with high fitnesses are likely to discovered anyway, as with the case with the maze-navigating robot.

In summary, indirect encodings are difficult to reproduce because of their hierarchical and repetitive nature. If SHRUTI or other neural-symbolic networks are to be represented and evolved using indirect encodings as inspired by nature, similar difficulties can be expected and are indeed encountered as shown in the results in chapter 4.

2.6. Evolution of neural-symbolic networks

The developers of SHRUTI argue that the probability of learning SHRUTI's structures from a randomly interconnected network of neurons is small, and that neural structures would require preorganisation in order to ensure that facts, relations and other concepts can be learned [96, 120]. A SHRUTI network contains multiple instances of the same structures, perhaps with some variation such as different numbers of arguments in predicates and facts. Repeated neural structures are common in biology [103], in which they may be defined once in the genome but expressed multiple times. SHRUTI's developers argue that SHRUTI networks are well suited to such representations and therefore indirect encoding of SHRUTI networks would support the biological plausibility of the SHRUTI model. This thesis presents the first genotypic indirect encoding for SHRUTI networks and explores the biological plausibility of such genomes even further by testing their evolvability.

Artificial development can be used to evolve generative and development systems that represent phenotypes using indirect encoding, inspired by DNA [13, 105]. Neural networks are among the systems that can be represented and evolved in this way, and results demonstrate that particular structures encoded only once in the genome may be expressed multiple times [39, 56, 85, 106]. Not only is such an approach biologically plausible but it allows representations to be scalable in that the size of the phenotype is not proportional to that of the genotype and the same genome can be used to produce networks of different sizes. For these reasons, artificial development and generative and developmental systems provide a suitable solution for the representation and evolution of SHRUTI networks. However, generative and development systems are difficult to reproduce through objective-based evolution due to their hierarchical and repetitive nature [122], and the same is later observed to be true of developmental SHRUTI networks in chapter 4.

In addition to testing the claims of SHRUTI's developers, representing and evolving SHRUTI networks in this way contributes towards another goal of evolving neural networks: that of discovering neurally plausible models of intelligence [55, 71, 85, 106]. Artificial development has been used to evolve neural controllers for a range of tasks [37, 38, 39, 47, 55, 106]. If research is to progress towards evolving a general model of cognition, cognitive reasoning must be included in this model, and this thesis explores for the first time whether or not this is possible. Successful artificial development of SHRUTI networks may extend to other neural-symbolic models that allow for more expressive cognitive reasoning capabilities, especially since many of them relate to the two or three-layer general relational structure (section 2.3.3), albeit in different ways. Furthermore, evolving neural-symbolic models of reasoning may produce models that aid our understanding of how reasoning is performed at the neural level and not just at the higher anatomical level. Finally, due to similarities such as Hebbian and associative learning, successful evolution of neural-symbolic networks may even contribute to the evolution of neural networks for associative learning tasks in general.

3. Evolution with Hebbian learning

In testing the evolvability of SHRUTI networks, the first step was to attempt the evolution of one of the simplest and most fundamental of its structures: relations between predicates. This chapter introduces genomes that provide scalable instructions for developing connections between neurons[1] in SHRUTI networks so that the networks can learn logical relations based on evidence provided by a series of event observations (section 3.2). The results of evolving similar genomes are also presented (sections 3.3 and 3.4). The key findings of the research presented in this chapter can be summarised as follows. It was indeed possible to represent SHRUTI relations of the form $A \rightarrow B$ using a developmental genome that was scalable to logic programs of different sizes, and it was also possible to discover such genomes through an evolutionary search. Both findings support the biological plausibility of the SHRUTI model and the claim of SHRUTI's developers that SHRUTI networks can be represented using indirect encoding [96, 120]. The following paragraphs provide a more detailed overview of the chapter.

Before any investigation into SHRUTI genomes and their evolvability could begin, a deeper analysis of SHRUTI's causal Hebbian learning algorithm was required, as the original literature only details tests on small logic programs with limited expressivity [118]. Larger logic programs were required in order to demonstrate the scalability of the learning algorithm and the genomes evolved to support it. The findings of this analysis are described in section 3.1 and show that the ability of the causal Hebbian learning algorithm to learn probabilistic relations is limited in that it cannot learn the probabilities for pairs of relations of the form $P \rightarrow Q$, $\neg P \rightarrow R$. A solution is presented that is sufficient for learning the correct truth values of target relations, but not the probabilities.

A model of indirect encoding for SHRUTI networks is then presented in section 3.2, along with examples of scalable genomes constructed from the model that successfully develop SHRUTI networks capable of answering all of their questions correctly. These findings show that, as SHRUTI's developers suggest [96, 120], the biological plausibility of the SHRUTI model is supported by the fact that it can be represented using indirect encoding. The scalability of the genomes to programs of different sizes is owed to the fact that rather than representing a network for a particular logic program, each genome represents a single relational structure for two arbitrary predicates which can be constructed for each relation in the target logic program.

Section 3.3 describes the methodology used to evolve genomes that adhere to the genome model and section 3.4 presents the findings. Genomes which develop SHRUTI networks

[1] The development of the neurons themselves is addressed in chapter 4.

were successfully evolved, thus demonstrating that further biological plausibility can be ascribed to the fact that SHRUTI genomes for relational structures can be discovered through the biologically plausible means of evolution. The majority of genomes evolved were scalable to logic programs of different sizes and generalisable to logic programs supported by event sequences other than the sequences those genomes were evolved on. The behaviours of all the evolved genomes that produced working SHRUTI networks were very similar in that they all employed the same strategy for constructing those networks: formation of connections between neurons of the same function. Occasionally the genomes enforce additional conditions that reflect the behaviour of the Hebbian learning algorithm, such as restricting the formation of connections to active neurons or to neurons of equal phases. As will be explained in greater detail in section 3.2.4, this suggests that reducing the number of connections in a SHRUTI network using indirect encoding is only possible by incorporating elements of the learning algorithm into the rules encoded by the developmental genome, thus rendering the learning algorithm almost redundant.

Experiments were all conducted on logic programs containing predicates with no more than three arguments, and section 3.5 argues that SHRUTI genomes are scalable to relations containing larger predicates, with some results to support this case. In particular, the evolved genomes can develop relational structures that enable up to seven variable bindings to be performed simultaneously, thought to be roughly the number of variable bindings which a human can perform at any one time [95].

Finally, a discussion elaborating on the findings of this chapter and how they support the arguments made above is presented in section 3.6.

Throughout this chapter, all experiments are performed on simple SHRUTI networks in which each node is composed of one neuron, as opposed to an ensemble of neurons. Therefore the terms ‘node’ and ‘neuron’ will be used interchangeably with respect to SHRUTI networks henceforth.

3.1. Hebbian learning

In the SHRUTI literature [118], causal Hebbian learning was only tested on very small logic programs that did not include predicate arguments, as shown in figure 2.12 on page 50. Furthermore, relational structures did not include mediator clusters and instead connected antecedent and consequent nodes directly. Development of SHRUTI networks for such simple test scenarios is insufficient to properly demonstrate the scalability of indirect encodings for SHRUTI networks. Therefore larger, more expressive data sets were produced for the experiments conducted in this chapter. Unlike the original test scenario in [118], these logic programs contained predicate arguments, and negation of predicates was possible. Experiments with and without negated predicates were performed separately because it would later become apparent that learning relations containing negated relations is problematic for SHRUTI’s Hebbian learning algorithm (section 3.1.2). Mediator structures were excluded from the original SHRUTI learning experiments because the problem

of applying Hebbian learning to them had not been solved. Therefore these, and conjunctive relations which require intermediary nodes, are excluded from these experiments also but are addressed in chapter 4, in which a Hebbian learning mechanism for mediator structures is proposed.

Although this section demonstrates that causal Hebbian learning can be extended to predicates, the maximum number of predicate arguments in any logic program used as a test scenario is restricted to three. Therefore no more than three variable-bindings are performed at any one time, even though the human brain is believed to be capable of performing 7 ± 2 bindings at any one time [95]. A maximum of three arguments was chosen in order to reduce the size of the logic program and therefore the network required to represent it. This decision was made in order to reduce execution time in later experiments when measuring the fitness of evolved networks in section 3.4. The largest network in which logic programs can be learned is one in which each pair of nodes that perform the same function share a connection in both directions (the prerequisite for learning in [118]). Therefore the total number of connections between argument nodes could be as high as N^2 (where N is the total number of argument nodes), and the time taken to evaluate a network is at most $O(n^2)$. Thus, in order to reduce the number of connections to be evaluated when evaluating networks during evolution, it was necessary to limit the number of predicate arguments in each predicate. However it is argued later in section 3.5 that SHRUTI genomes will scale to any number of arguments, with some preliminary experiments to demonstrate scalability to a relation (but not an entire logic program) containing seven predicate arguments in each predicate. A more thorough investigation is a matter for future work.

All logic programs listed in sections 3.1.1 and 3.1.2 can be learned from sets of fixed and probabilistic event sequences listed in appendix A. As in the original SHRUTI learning experiments, the prior probability of a predicate is set to determine the frequency at which that predicate is observed at any point in time, and the probability associated with a relation is set to determine the frequency at which the consequent is observed after an occurrence the antecedent ($P(consequent|antecedent)$). Although in most cases the learned connection weights closely reflect the probabilities according to the evidence presented[2], the priority of this work was to develop networks that learn relations, with less concern for the probabilities. In other words, the aim was to learn whether or not relations were true or false based on observed evidence. A relation is regarded as true when the connection weights in the structure representing it are 0.5 or greater, strong enough to break the threshold of target nodes. For all logic programs, fixed event sequences and probabilities (both prior and relational) were set so that the strength of learned connection weights was sufficient to reflect the truth of the target relations. Although one might argue that fixed sequences and probabilities are defined to force a desired outcome, this is necessary because the aim is to simulate sequences of events that would lead to the target relations being learned by a human observer in a real-life scenario, and also for the SHRUTI network and its Hebbian learning algorithm to simulate this behaviour. In some event sequences, the evidence for certain relations or predicates was defined to

[2]Logic programs containing negation are an exception, for reasons explained further in section 3.1.2

appear more frequently than that of others in cases where relations may be ambiguous in the event sequence. For example, in the probabilistic event sequence for the NoNeg2 logic program (fig. 3.1) the prior probability of $R(y, z)$ was defined to be greater than that of other predicates (0.5 as opposed to 0.2). Whenever $P(x, y, z)$ is observed, $Q(x, y)$ and $R(y, z)$ are observed as consequents, and $S(x, y)$ follows as a consequence of $Q(x, y)$. However, in this particular instance $S(x, y)$ follows both $Q(x, y)$ and $R(y, z)$. Therefore although the target relation $Q(x, y) \rightarrow S(x, y)$ gains strength, $R(y, z) \rightarrow S(x, y)$, which is not a target relation, also gains strength. $R(y, z) \rightarrow S(x, y)$ is not true in the target NoNeg2 program, and therefore further evidence must be presented in order to weaken it. In order to do this, $R(y, z)$ must be observed to occur more frequently without $S(x, y)$ than with, and therefore the prior probability of $R(y, z)$ was set to be greater so that the number of instances for which $R(y, z)$ occurs alone was also greater.

For each logic program, two sets of questions are also defined: an A set and a B set, which are also listed in appendix A. The A set contains one question for every relation in the logic program and will be used in the fitness function for measuring the performance of evolved networks later in section 3.3.3, as it was hypothesised that one question per relation was the minimum required to assess the efficacy of an evolved network. The reasoning behind this hypothesis is explained further in section 3.3.3. The B set contains every question expected to be answered 'true' or 'false' for each logic program and is used to demonstrate the efficacy of the learning algorithm in sections 3.1.1 and 3.1.2. Both question sets contain one question expected to be answered 'unknown' for each predicate in order to demonstrate that networks do not yield the correct answer by simply always answering 'true' or 'false' for any given predicate, and to discourage the evolutionary search (section 3.4) from discovering networks that do so.

A window of synchrony of two observations was used when training all networks. In other words, if an instance of predicate P is observed and an instance of Q is then observed within the space of two more time steps, the relation $P \rightarrow Q$ is strengthened. Otherwise, if Q is observed later than two time steps after P (or not at all), the relation is weakened.

3.1.1. Logic programs without negation

Fig. 3.1 presents four new logic programs varying in size that do not include negated predicates. Fig. 3.2 shows the results of running SHRUTI's causal Hebbian learning algorithm on each data set and demonstrates that the algorithm is suitable for logic programs of these sizes. As in the original SHRUTI test scenario for Hebbian learning, all nodes are connected by function (e.g. enabler-to-enabler) with initially weak weight values and presented a sequence of 500 event observations[3] generated according to a set of probabilities. The performance of the network is tested on each network's B set: the set of all truths that can be inferred from the facts and relations in the logic program (excluding the facts themselves), along with one question per predicate which the logic program is unable to answer. For each logic program, the connection weights eventually settle in such

[3]This includes null observations, i.e. time steps in which no event observations occur.

NoNeg1

Relations	Facts
$P(x,y) \rightarrow Q(x,y)$	$P(a,b)$
$Q(x,y) \rightarrow R(x,y)$	$P(c,d)$
	$Q(e,f)$

NoNeg2

Relations	Facts
$P(x,y,z) \rightarrow Q(x,y)$	$P(a,b,c)$
$P(x,y,z) \rightarrow R(y,z)$	$Q(d,e)$
$Q(x,y) \rightarrow S(x,y)$	$R(f,g)$

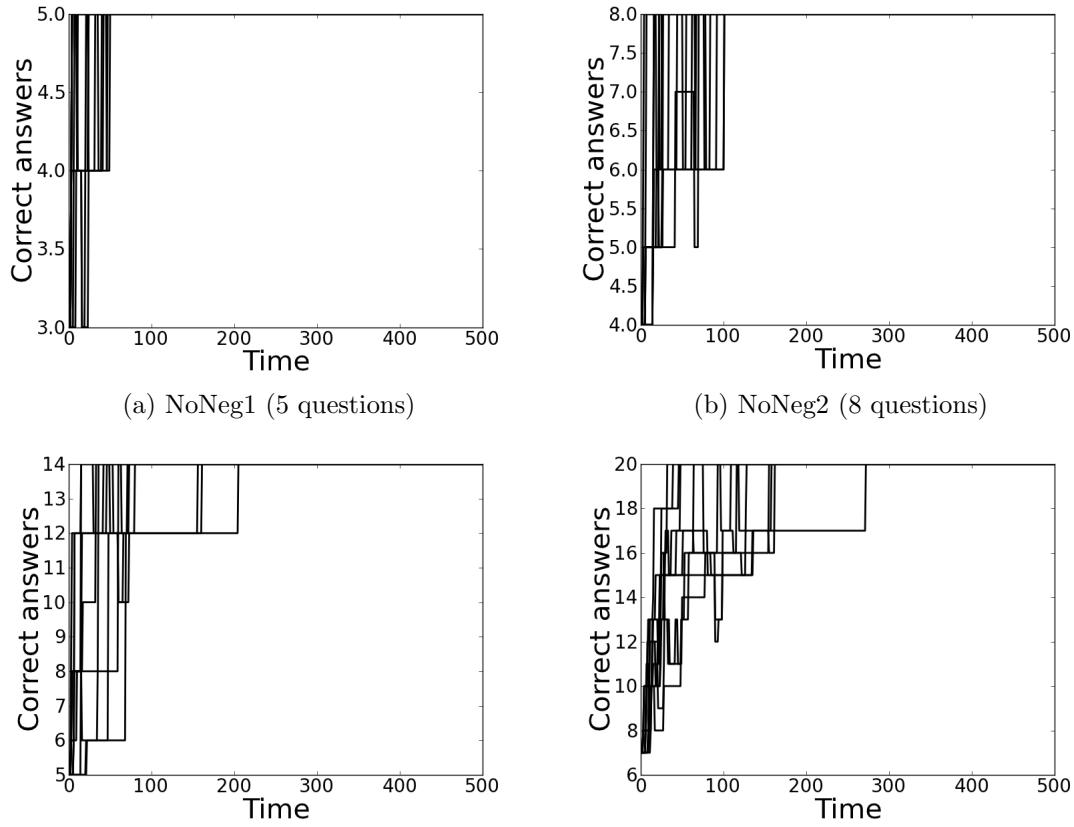
NoNeg3

Relations	Facts
$P(x,y,z) \rightarrow R(x,y)$	$P(a,b,c)$
$Q(y,x) \rightarrow R(x,y)$	$Q(d,e)$
$Q(y,x) \rightarrow S(y,x)$	$Q(f,g)$
$R(x,y) \rightarrow T(x)$	$R(h,i)$

NoNeg4

Relations	Facts
$P(x,y) \rightarrow Q(x,y)$	$P(a,b)$
$Q(x,y) \rightarrow R(x)$	$Q(c,d)$
$Q(x,y) \rightarrow S(x,y)$	$S(e,f)$
$S(x,y) \rightarrow T(y)$	$U(g,h,i)$
$U(x,y,z) \rightarrow V(x,y,z)$	$V(j,k,l)$
$V(x,y,z) \rightarrow S(x,y)$	

Figure 3.1. Logic programs without negated predicates, created for testing the causal Hebbian learning algorithm (section 3.1) and developmental genomes (section 3.2) on larger logic programs than were used in the literature.



(a) NoNeg1 (5 questions)

(b) NoNeg2 (8 questions)

(c) NoNeg3 (14 questions)

(d) NoNeg4 (20 questions)

Figure 3.2. The number of correct answers to questions over time for each NoNeg data set. 10 trials are performed for each logic program, generating a new event sequence for each trial.

a way that the corresponding network can answer all of its questions correctly (figure 3.2). Furthermore, as was the case in the original SHRUTI test scenario (table 2.12, page 50), learned connection weights reflect the probabilities of the corresponding relations. This is shown for NoNeg4 in table 3.1.

Relation	Probability	Connection	Average weight
$P(x,y) \rightarrow Q(x,y)$	0.9	+P → +Q ?Q → ?P	0.842 0.831
$Q(x,y) \rightarrow R(x)$	0.8	+Q → +R ?R → ?Q	0.795 0.769
$Q(x,y) \rightarrow S(x,y)$	0.75	+Q → +S ?S → ?Q	0.710 0.731
$S(x,y) \rightarrow T(y)$	0.85	+S → +T ?T → ?S	0.835 0.836
$U(x,y,z) \rightarrow V(x,y,z)$	0.7	+U → +V ?V → ?U	0.709 0.690
$V(x,y,z) \rightarrow S(x,y)$	0.8	+V → +S ?S → ?V	0.720 0.734

Table 3.1. Final weights of connections representing relations in NoNeg4, following training on probabilistic event sequences. The learned weight values are averaged over 10 trials, and are similar to the probabilities of the corresponding relations.

3.1.2. Logic programs with negation

Figure 3.3 presents four new logic programs that do contain negated predicates. Attempting to learn these in a SHRUTI network revealed that SHRUTI's causal Hebbian learning algorithm is limited in that it cannot always learn logic programs such as these. After attempting to learn the Neg data sets, none of the networks were able to answer all their questions correctly, as shown in figure 3.4. While most learned connection weights reflect the probabilities of the corresponding relations, this is not the case for enabler-to-enabler connections in certain relations. Table 3.2 shows that the weights for the connections ?S → ?Q and ?T → ?Q in Neg4 converge at 0.529 and 0.287 respectively, when they should both converge at around 0.85. In the case of ?T → ?Q, the connection weight (0.287) is insufficient to activate ?Q during inference, and therefore queries on the relation $\neg Q(y,z) \rightarrow \neg T(y,z)$ cannot be answered. Therefore not all questions in Neg4 can be answered correctly, and a similar problem is encountered in each logic program.

The problem can be explained as follows. In each Neg logic problem, there exists a predicate for which the truth and falsity both act as antecedents in different relations, and the

Neg1

Relations	Facts
$P(x,y) \rightarrow Q(x,y)$	$P(a,b)$
$\neg P(x,y) \rightarrow \neg R(x)$	$\neg P(c,d)$

Neg2

Relations	Facts
$P(x,y,z) \rightarrow \neg Q(x,y,z)$	$P(a,b,c)$
$Q(x,y,z) \rightarrow \neg R(y,z)$	$Q(d,e,f)$
$\neg Q(x,y,z) \rightarrow S(x,y)$	$\neg Q(g,h,i)$

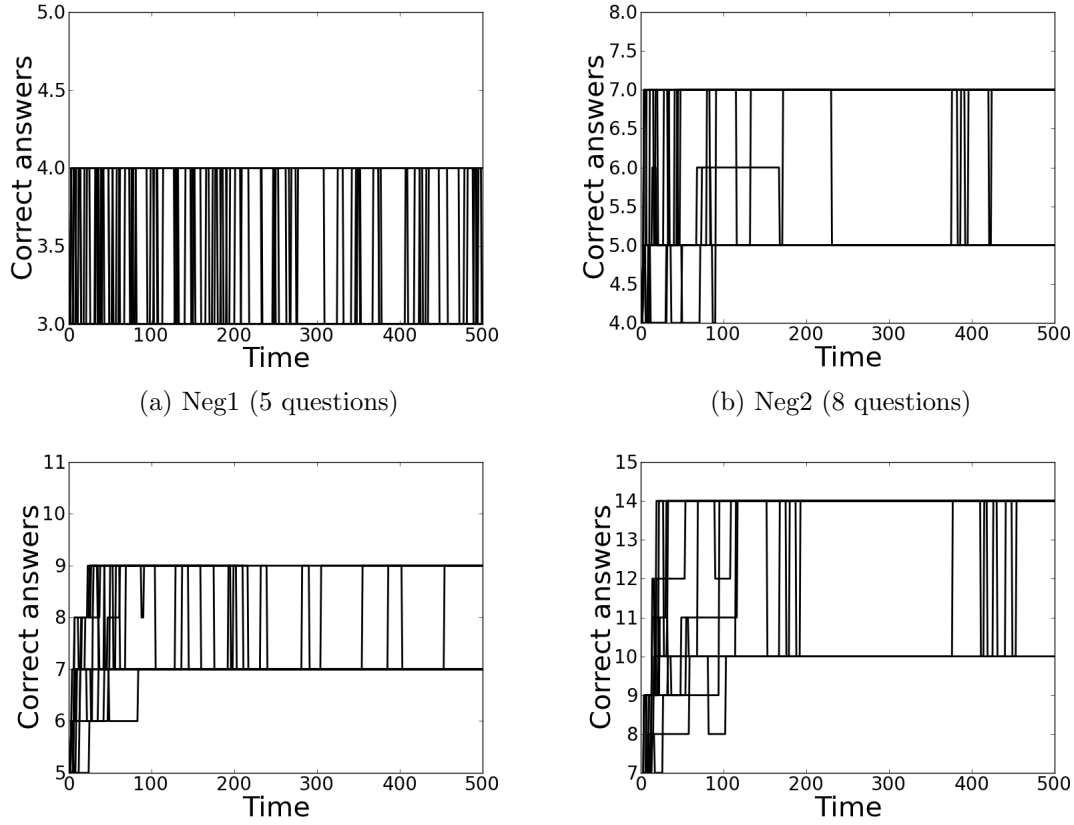
Neg3

Relations	Facts
$P(x,y) \rightarrow \neg Q(y,x)$	$P(a,b)$
$\neg P(x,y) \rightarrow \neg R(x,y)$	$\neg P(c,d)$
$\neg Q(y,x) \rightarrow S(x,y)$	$\neg Q(e,f)$
$\neg R(x,y) \rightarrow T(y)$	$\neg R(g,h)$

Neg4

Relations	Facts
$\neg P(x,y,z) \rightarrow Q(y,z)$	$\neg P(a,b,c)$
$Q(y,z) \rightarrow S(y,z)$	$Q(d,e)$
$\neg Q(y,z) \rightarrow \neg T(y,z)$	$\neg Q(f,g)$
$\neg R(z,x,y) \rightarrow S(y,z)$	$\neg R(h,i,j)$
$S(y,z) \rightarrow \neg U(y)$	$S(k,l)$

Figure 3.3. Logic programs with negated predicates, created for testing the causal Hebbian learning algorithm (section 3.1) and developmental genomes (section 3.2) on larger logic programs than were used in the literature.



(a) Neg1 (5 questions)

(b) Neg2 (8 questions)

(c) Neg3 (11 questions)

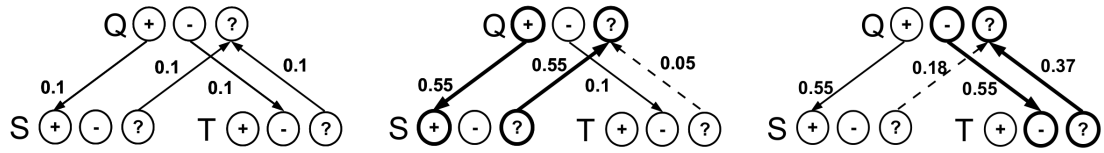
(d) Neg4 (15 questions)

Figure 3.4. The number of correct answers to questions over time for each Neg data set. In none of the 10 trials performed were the networks able to adjust connection weights such that they could answer all questions correctly. Instead, the number of correct answers oscillates.

Relation	**Probability**	**Connection**	**Average weight**
$\neg P(x,y,z) \rightarrow Q(y,z)$	0.9	-P $\rightarrow$ +Q	0.847
		?Q $\rightarrow$?P	0.851
$Q(y,z) \rightarrow S(y,z)$	0.85	+Q $\rightarrow$ +S	0.759
		?S $\rightarrow$?Q	**0.529**
$\neg Q(y,z) \rightarrow \neg T(y,z)$	0.85	-Q $\rightarrow$ -T	0.84
		?T $\rightarrow$?Q	**0.287**
$\neg R(z,x,y) \rightarrow S(y,z)$	0.8	-R $\rightarrow$ +S	0.741
		?S $\rightarrow$?R	0.749
$S(y,z) \rightarrow \neg U(y)$	0.9	+S $\rightarrow$ -U	0.90
		?U $\rightarrow$?S	0.91

Table 3.2. Final weights of connections representing relations in Neg4, following training on probabilistic event sequences. The learned weight values are averaged over 10 trials, and in most cases are similar to the probabilities of the corresponding relations. However, owing to the *conflicting relations problem* (fig. 3.5), this is not the case for ?S $\rightarrow$?Q and ?T $\rightarrow$?Q, shown in bold.

algorithm struggles to learn both of these relations. For example, in Neg4, a consequence exists for the truth and falsity of $Q(y,z)$ ($Q(y,z) \rightarrow S(y,z)$ and $\neg Q(y,z) \rightarrow \neg T(y,z)$). When $Q(y,z)$ and $S(y,z)$ are observed to coincide, the corresponding connection weights increase as expected (fig. 3.5b). However, connections also exist between the enabler and role nodes of $Q(y,z)$ and those of $T(y,z)$. The enabler and role nodes of $Q(y,z)$ are active because of the observation that has just been made, but those of $T(y,z)$ are not, and therefore the connections between them weaken according to the rules of causal Hebbian



(a) Before learning (b) Learning $Q \rightarrow S$ (c) Learning $\neg Q \rightarrow \neg T$

Figure 3.5. The *conflicting relations problem*. When evidence for $Q \rightarrow S$ is observed, $?Q$ is activated with no activation at $?T$, and so the connection between these two nodes weakens. When evidence for $\neg Q \rightarrow \neg T$ is observed, the connection between $?Q$ and $?S$ weakens. Connections being strengthened are shown in bold, and connections being weakened are shown as broken lines.

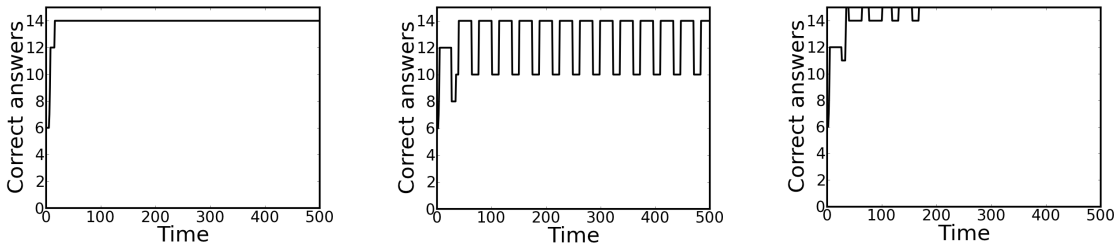
learning (fig. 3.5b). When $\neg Q(y, z)$ and $\neg T(y, z)$ are observed to coincide, the reverse is true; connections between $Q(y, z)$ and $T(y, z)$ strengthen but the enabler and role nodes connections between $Q(y, z)$ and $S(y, z)$ weaken (fig. 3.5c). In summary, pairs of relations of the form $Q \rightarrow S$ and $\neg Q \rightarrow T$ cannot be learned without causing each other's enabler-to-enabler and role-to-role connections to weaken. As a consequence, these conflicting relations are repeatedly learned and unlearned, and the number of questions the network can answer correctly oscillates. This problem will be referred to as the *conflicting relations problem* henceforth.

To resolve this problem, $Q(y, z)$ and $\neg Q(y, z)$ must be observed an equal number of times with a bias towards strengthening weights in order for both relations to be learned. As connection weights for one relation get stronger, weights for the other get weaker, and therefore observing evidence for both an equal number of times ensures that the weights of both relations are balanced. Setting a bias towards the strengthening of weights ensures that the value at which these balanced weights converge is above 0.5, strong enough to activate target nodes so that both relations can be interpreted as true. To enforce the necessary bias towards strengthening weights, two new variables $hInc$ and $hDec$ were introduced to adjust the learning rate α so that it may be different when strengthening (equation 2.1, page 49) and weakening (equation 2.2) weights. α is now defined as shown in equation 3.1.

$$\alpha = \begin{cases} hInc/\#Updates & \text{when strengthening weights} \\ hDec/\#Updates & \text{when weakening weights} \end{cases} \tag{3.1}$$

The experiment for Neg4 was repeated with the fixed event sequence, in which $Q(y, z)$ and $\neg Q(y, z)$ are observed an equal number of times. $hInc$ was set to 1.25 and $hDec$ was set to 1 in order to enforce the required bias towards strengthening relations. Fig. 3.6c shows that all questions can be answered correctly under this implementation. Training with the probabilistic event sequence (in which the observations are not guaranteed to occur an equal number of times) has a negative effect on the learning algorithm (fig. 3.6a), as does removing the bias by setting $hInc$ to 1 (fig. 3.6b).

Because of the conflicting relations problem, relations are learned and unlearned until evidence for each has been observed enough times to support evidence for both. Therefore the correct answering of any questions that depend on conflicting relations are periodi-



(a) Probabilistic event sequence with bias.

(b) Fixed event sequence with no bias.

(c) Fixed event sequence with bias.

Figure 3.6. Overcoming the conflicting relations problem for Neg4. All 15 questions can only be answered correctly when trained on a fixed event sequence, with a bias towards strengthening weights.

cally answered correctly and incorrectly until stability has been achieved (fig. 3.6c). Such rising and falling fitness is similar to a phenomenon observed in developmental psychology known as *U-shaped development* in which as a child learns a cognitive skill, particularly in natural language development, he or she may show signs of decline before improving [79].

In summary, SHRUTI's causal Hebbian learning algorithm does work on logic programs larger than those presented in the literature [118] but cannot learn logic programs in which a predicate is negated in some relations but not others, unless it is trained on a fixed event sequence and with a bias towards strengthening relations. In all cases, collector-to-collector connections still reflect the probabilities associated with a target logic program accurately and the same is true for enabler-to-enabler and role-to-role connections when the logic program contains no conflicting relations. However when logic programs are learned for networks with conflicting relations, enabler-to-enabler and role-to-role connections do not reflect the probabilities, because it is the weights of these connections that suffer the oscillation during learning. For this reason, learned connection weights of the networks developed in this thesis are only claimed to accurately reflect the truth value of relations so that queries on the logic programs can be answered correctly, but are not claimed to always reflect the probabilities associated with those relations. Given that the ability of causal Hebbian learning to learn probabilistic relations has been shown to be limited, the reader is advised to refer to the literature on RTRBMs for a neural-symbolic model of probabilistic learning and reasoning [27, 28]. If one is interested in the learning of probabilistic relations in general and biological plausibility is not a concern, then such models also exist [54, 83].

Because both the Neg and NoNeg logic programs are capable of learning from the fixed event sequences, it was decided to use fixed event sequences for measuring objective fitness in the evolutionary experiments described later in sections 3.3 and 3.4. This carried additional advantages with regards to the complexity of the evolutionary process in that for each genome at each generation, exactly the same training data could be used and multiple trials of random event sequences for each genome would not be necessary. Furthermore, it was expected that most networks evolved on the fixed event sequences would generalise to the probabilistic sequences for the NoNeg logic programs. This would later be verified by

training evolved networks on random event sequences and measuring their performance against test questions (section 3.4.3). However it was expected that this would not be the case for Neg logic programs, which cannot be learned using probabilistic event sequences anyway. Because the conflicting relations problem is a limitation of the original learning algorithm and not of the developmental genome introduced in section 3.2, demonstrating that evolved genomes generalise to NoNeg logic programs was sufficient to support the argument that generalisable genomes capable of supporting SHRUTI's existing learning capabilities do exist and can be discovered through evolution.

Now that the performance of SHRUTI when learning larger, more expressive data sets had been demonstrated, the next step was to produce a genome for developing connections between nodes that allow target relations to be learned.

3.2. SHRUTI genome

The SHRUTI developers argue that the prerequisite structure required to enable the learning of relations can be realised through a model of genetic development [96, 120]. This section describes the first developmental genome for encoding SHRUTI networks in this way, extending work originally presented by the author in [112]. The genome model develops connections between nodes (neurons) in SHRUTI predicates that allow relations between those predicates to be learned. This early genome model only develops connections between nodes in the network and does not produce the neurons themselves, thus assuming that clusters of neurons representing predicates and circuits representing facts already exist in the network. The development of neurons is addressed in chapter 4.

3.2.1. Learning and development cycle

For each event observation at time t, the network undergoes a learning and development cycle as shown in algorithm 1. At each t all events are observed and any existing weights are adjusted according to SHRUTI's learning algorithm, as before. However before the next cycle begins, the developmental stage takes place. In this stage, the set of conditions defined in the genome are tested for each possible pair of neurons and any actions for which all conditions are met are executed. The action may either be the addition or removal of

Algorithm 1 Learning and development cycle

for each event observation t **do**

Learning stage: For each neuron pair, update connection weights according to rules of causal Hebbian learning described on page 49 (see also algorithm 2, appendix B).

Development stage: For each neuron pair, add or remove connection according to rules specified in the genome, to be interpreted as described in section 3.2.2 (see also algorithm 3, appendix B).

end for

a connection between two neurons. The genome for controlling development is described in detail in the following subsections.

3.2.2. Genome structure

The genome is a genetic program [7] that represents a decision tree of conditions and actions as shown in figure 3.7c. Each path from the root of the tree to a terminal node corresponds to a different rule in which the terminal node is an action to be performed and the nodes preceding it are conditions which must be met in order for that action to be performed. To use terminology from gene expression, a node in the tree is said to be *expressed* when it is encountered in the tree search.

The neuron for which an input is currently being considered is labelled as 'SELF' in the genome, and an input neuron is labelled 'P_INPUT' (potential input) if a connection to SELF from that neuron does not exist or 'E_INPUT' (existing input) if the connection does exist. Let $type(n)$ specify the role that a genotypic node n plays in the rule. That is, $type(n) \in \{SELF, P\_INPUT, E\_INPUT, ACTION\}$. There is a transitive ordering $\prec$ of types such that $SELF \prec P\_INPUT$, $SELF \prec E\_INPUT$, $P\_INPUT \prec ACTION$ and $E\_INPUT \prec ACTION$. In other words, conditions which affect SELF are considered first and therefore conditions on E_INPUT or P_INPUT cannot express conditions on SELF. This ordering was chosen in order to reduce execution time, because the truth value of any SELF condition in a particular rule will be the same regardless of which potential or existing inputs are also being tested by the same rule, and therefore it is unnecessary to continue testing the same SELF condition for every potential or existing input. Furthermore, E_INPUT and P_INPUT conditions cannot express each other since a neuron cannot be both an existing input and a possible input. All condition nodes can express action nodes.

Although the genome represents a tree structure, the genome itself takes the form of a string of elements that describes the structure of that tree, as shown in figure 3.7a. The genome contains a sub-string for each genotype node (condition or action) and each node is uniquely identified by the index of the corresponding sub-string. For each condition, the genome encodes the attribute to be tested, an operator (<,≤,= ≥,>,) and the value to test that argument against. The attributes that can be tested are listed in table 3.3 on page 81. Some attributes only apply to existing connections, i.e. to a connection between SELF and E_INPUT. For conditions testing P_INPUT or E_INPUT, the value an attribute is tested against may be set to 'SELF' in order to produce a condition that compares the attributes of two neurons. For example, $P\_INPUT.function = SELF.function$ is true if both SELF and P_INPUT perform the same function (enabler, collector, etc.). The next condition or action to express in the event of a condition being evaluated as true or false is also encoded in the genome. For example, condition 1 is represented by the first sub-string in the genome and can be interpreted as saying 'if the activity of SELF is over 0.5, then express condition (sub-string) 4, otherwise express condition (sub-string) 3'. An index of 0 terminates the tree search with no action expressed. For an action node, the

SELF.act	>	0.5	4	3	E_INPUT.nInputs	<	4	4	5		
(1) Condition					(2) Condition						
	E_IN.weight	<	0.1	6	0	P_IN.act	>	0.5	5	0	
	(3) Condition					(4) Condition					
	P_IN.func	=	SELF.func	7	0	DEL		ADD	0.1		
	(5) Condition					(6) Action		(7) Action			

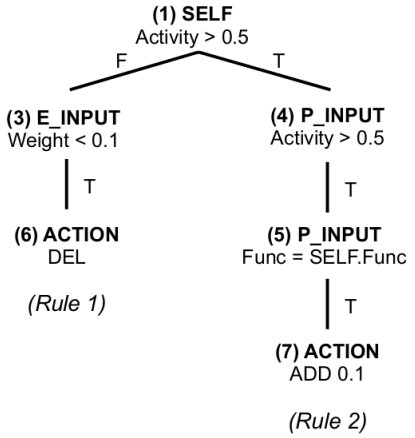
(a) Raw genome

1. If SELF.Activity >0.5: Go to 4, else 3
2. If E_INPUT.nInputs <4: Go to 4 else 5
3. If E_INPUT.Weight <0.1: Go to 6
4. If P_INPUT.Activity >0.5: Go to 5
5. If P_INPUT.Func = SELF.Func: Go to 7
6. ACTION: Delete connection
7. ACTION: Add connection with weight 0.1

Rule 1: Delete an existing input connection to SELF if its weight is below 0.1.

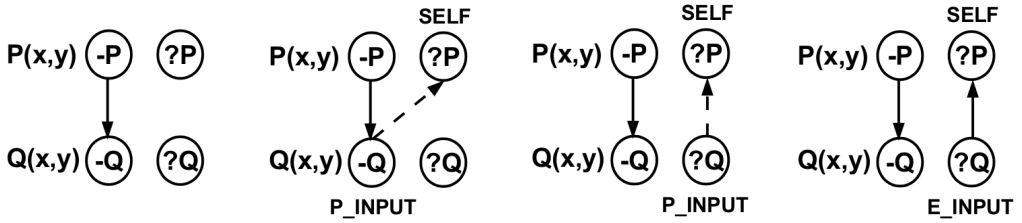
Rule 2: If SELF and a potential input are both active and perform the same function (enabler, collector, etc.), produce a connection with weight 0.1.

(b) Interpretation of conditions, actions and rules



(c) Decision tree

Figure 3.7. Genome for developing SHRUTI networks. The genome represents a decision tree in which each path from the root node to a terminal represents a rule. Non-terminal nodes represent conditions and terminal nodes represent actions to be performed when those conditions are satisfied. Actions either add or remove connections between two neurons. The neuron for which an input is being considered is labelled as 'SELF' in the genome, and an input neuron is labelled 'P_INPUT' if a connection from that neuron to SELF does not exist or 'E_INPUT' if the connection does exist. A complete explanation of the genome model can be found in the text.



(a) (b) (c) (d)

Figure 3.8. Example of rule 2 from the genome in fig. 3.7 constructing connections between enabler nodes in a SHRUTI network when predicate $Q(x, y)$ is observed to follow predicate $P(x, y)$. When considering potential inputs (P_INPUT) to ?P, the genome first considers $-Q$, but no connection is formed because $-Q$ performs a different function (collector) from $?P$ (enabler) and so condition 5 is not satisfied. $?Q$ does perform the same function and therefore meets the criteria for connection formation set by rule 2: both neurons must be active (conditions 1 and 4) and of the same function (condition 5). An input connection can therefore be formed by action 7, the terminal of rule 2.

Neuron conditions	
Activity	Activity level
Phase	Current phase
Time_Window	Remaining time before window of synchrony expires
Function	Neuron function: collector, enabler or role node.
nInputs	Total number of inputs
nActInputs	Total number of 'active' inputs, that is, those with weight above 0.5 and therefore strong enough to activate neurons.
Connection conditions (E_INPUT only)	
Weight	Weight of connection between SELF and E_INPUT
nUpdates	Number of times connection weight has been updated

Table 3.3. Attributes that may be tested by conditions in the genome.

Actions	
ADD (WEIGHT)	Add connection from P_INPUT to SELF with weight as specified by the parameter WEIGHT
DEL	Delete connection between SELF and P_INPUT

Table 3.4. Actions that may be expressed by the genome

type of action to be performed is specified (table 3.4, page 81). Two types of action are possible: the addition or removal of a connection between two nodes[4]. If the action to perform is the addition of a new connection, a parameter for specifying the weight of the new connection is also provided. For example, action 7, represented by the final sub-string in the genome, tells SELF to form an input connection of weight 0.1 from P_INPUT.

The genome may contain conditions or actions that are not expressed by any condition in the genome and are therefore unexpressable. For example, condition 2 is encoded in the genome (fig. 3.7a) but is not expressed in the decision tree (fig. 3.7c). This is analogous to the sequences of non-coding genes known as *introns* in DNA (section 2.1.6). Unexpressed conditions and actions will therefore also be referred to as *introns* henceforth, and expressed conditions and actions will be referred to as *exons*.

Fig. 3.8 provides an example of the genome in fig. 3.7 developing a connection between enablers. The genome contains two rules: one for removing connections that become weak after a number of updates (Rule 1) and another for forming connections between active neurons of the same function (Rule 2). A connection from $-P$ to $-Q$ already exists, but not yet from $?Q$ to $?P$ (fig. 3.8a). The genome in $?P$ begins to search for possible input connections, beginning with $-Q$ (fig. 3.8b). Both neurons are active, satisfying conditions 1 and 4, but do not satisfy condition 5 since the two neurons perform different functions. The genome then tests $?Q$ as a possible input and finds that conditions 1,4 and 5 are all satisfied since both neurons are active and perform the same function (enabler) (fig. 3.8c). A connection from $?Q$ to $?P$ can therefore be formed (fig. 3.8d).

[4]The addition and removal of neurons is addressed in chapter 4.

3.2.3. Testing the genome model

To test the genome model and its ability to develop SHRUTI networks, four genomes were created, labelled G1 to G4 (fig. 3.9). Each constructs connections between neurons of the same function. G1 does only this so that every neuron is connected to every other neuron that performs the same function and therefore a relational structure exists for every possible relation. G2 creates new inputs to SELF from every other neuron of the same function when SELF is active and removes redundant connections when inactive. G3 restricts the development of connections to when both SELF and P_INPUT are active but does not remove redundant connections, whereas G4 does both.

Each genome was tested on developing networks for logic programs NoNeg1-4 (fig. 3.1, page 73) and Neg1-4 (fig. 3.3, page 74) to demonstrate scalability and the ability to work

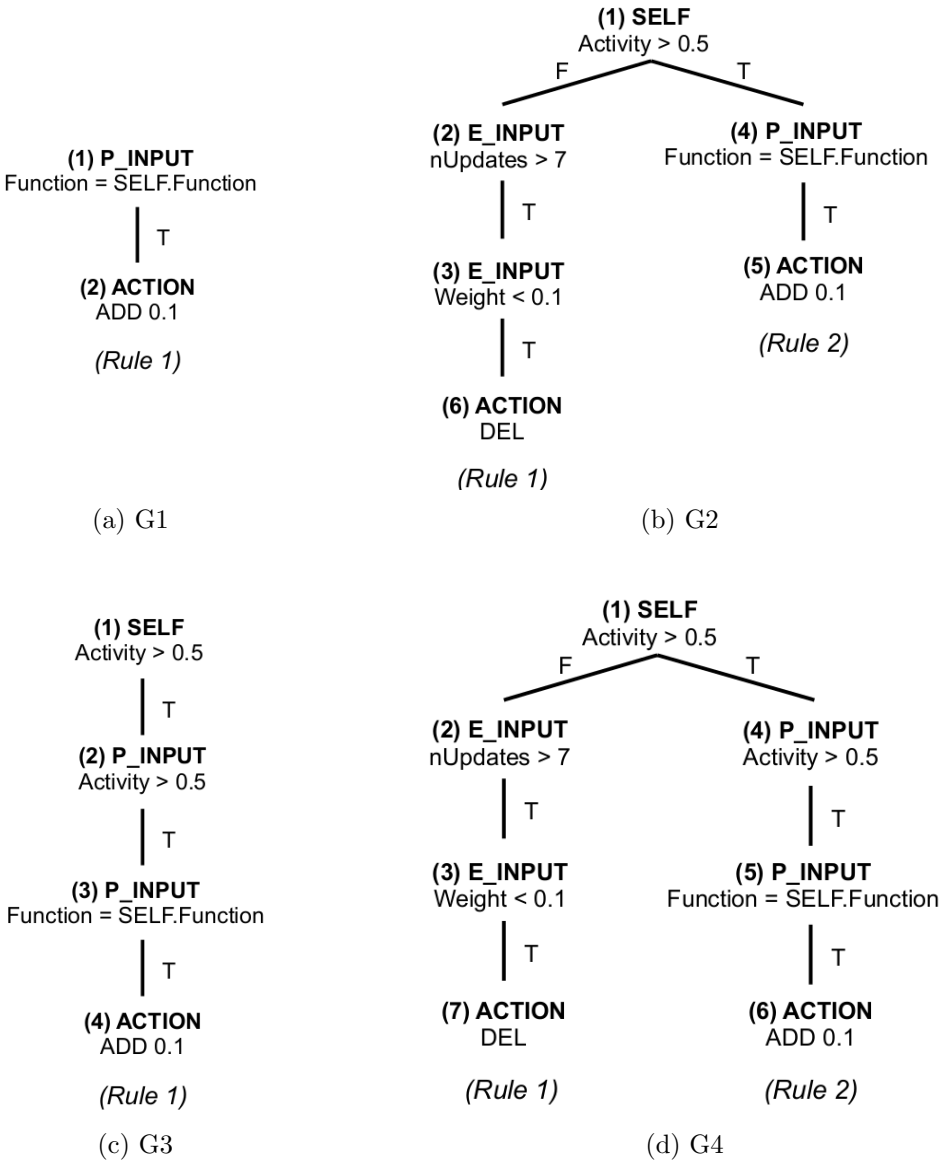
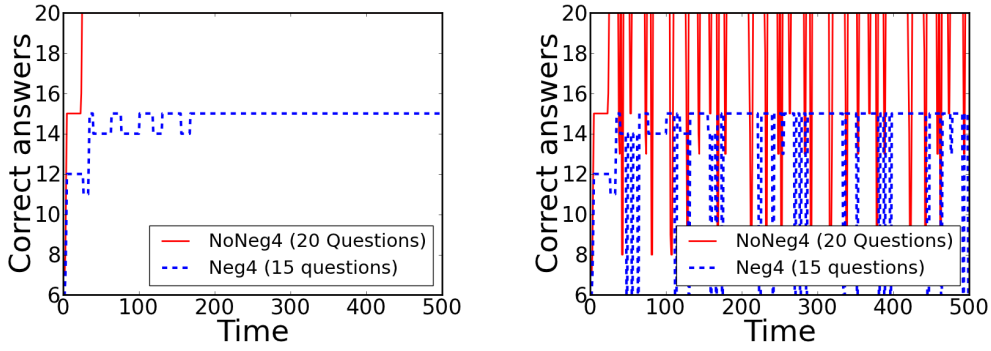
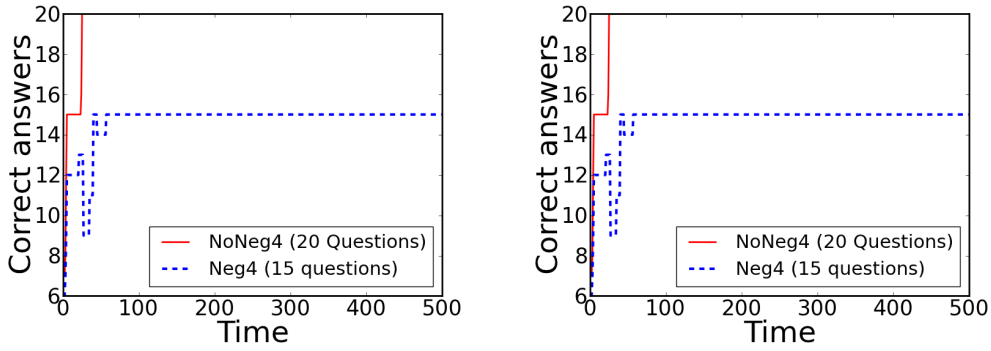


Figure 3.9. Genomes for developing working SHRUTI relations. G1 connects all neurons of the same function. G2 does this but prunes connections that are significantly weak after 7 weight updates. G3 only produces connections between active neurons of the same function. G4 does the same but also prunes weak connections.

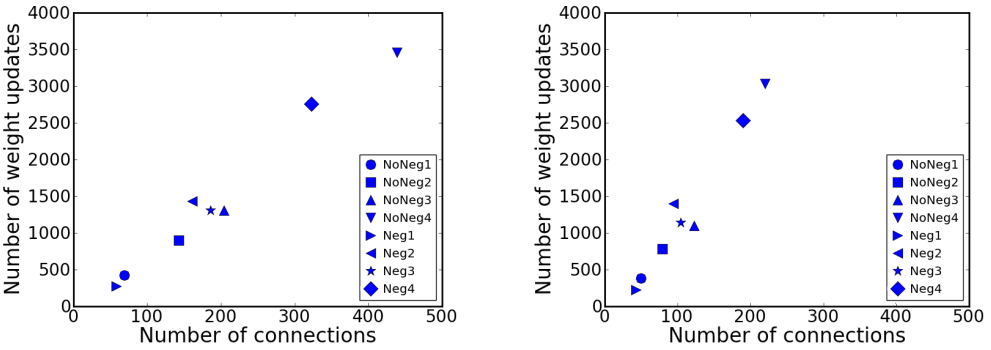


(a) G1 (b) G2

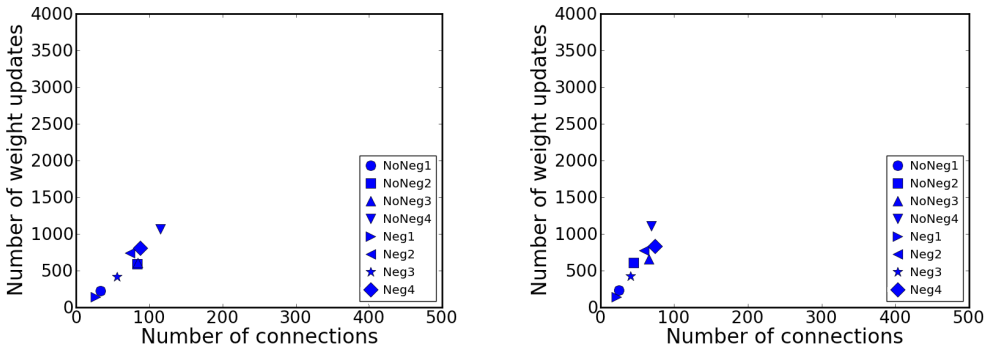


(c) G3 (d) G4

Figure 3.10. Fitness for each genome (measured as the number of correct answers) as they develop networks for NoNeg4 and Neg4 based on fixed event sequences.



(a) G1 (b) G2



(c) G3 (d) G4

Figure 3.11. The final number of connections and weight updates yielded by each genome when developing networks for all eight logic programs.

G1: Neurons all connected by function

Program	#Relations	#Predicates	#Conns	#Add	#Delete	#Updates
NoNeg1	2	3	69	66	0	426
NoNeg2	3	4	143	140	0	904
NoNeg3	4	5	204	200	0	1308
NoNeg4	6	7	439	434	0	3455
Neg1	2	3	58	56	0	272
Neg2	3	4	161	158	0	1435
Neg3	4	5	186	182	0	1308
Neg4	5	6	323	318	0	2758

G2: Neurons all connected by function, redundant connections removed

Program	#Relations	#Predicates	#Conns	#Add	#Delete	#Updates
NoNeg1	2	3	50	65	18	382
NoNeg2	3	4	79	130	54	787
NoNeg3	4	5	123	189	70	1104
NoNeg4	6	7	220	467	252	3035
Neg1	2	3	43	46	5	224
Neg2	3	4	94	206	115	1404
Neg3	4	5	104	181	81	1142
Neg4	5	6	190	385	200	2533

G3: Development restricted to active neurons of the same function

Program	#Relations	#Predicates	#Conns	#Add	#Delete	#Updates
NoNeg1	2	3	33	30	0	225
NoNeg2	3	4	83	80	0	592
NoNeg3	4	5	84	80	0	612
NoNeg4	6	7	115	110	0	1067
Neg1	2	3	26	24	0	139
Neg2	3	4	73	70	0	741
Neg3	4	5	56	52	0	413
Neg4	5	6	87	82	0	806

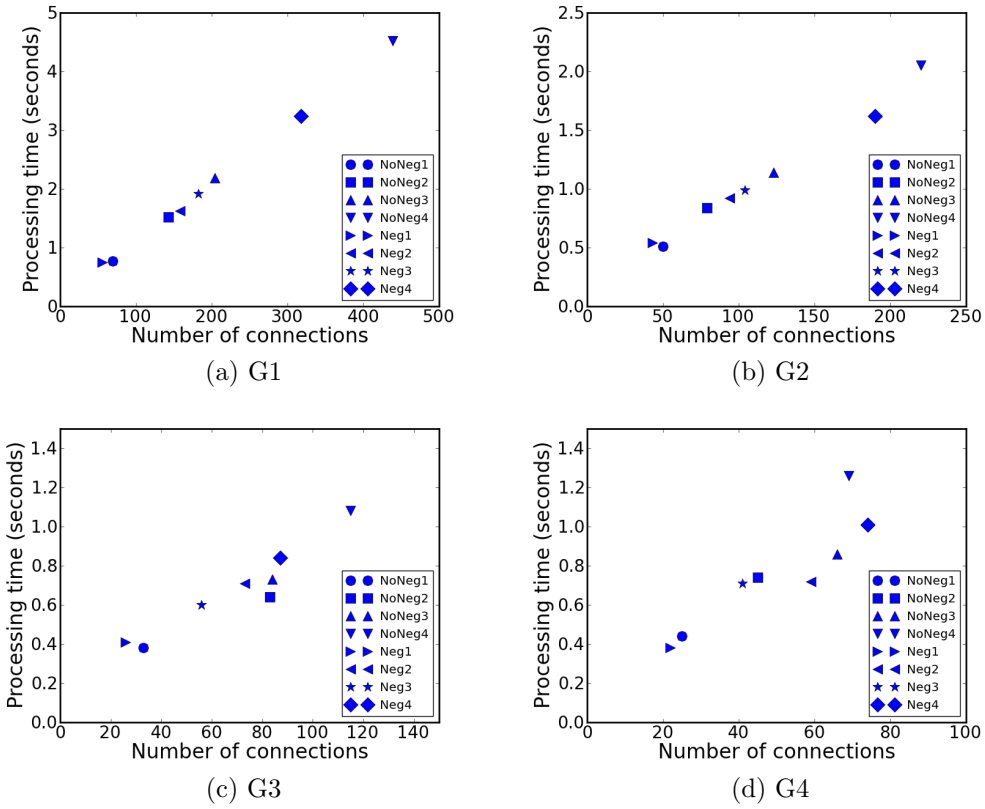
G4: Active neurons of the same function, redundant connections removed

Program	#Relations	#Predicates	#Conns	#Add	#Delete	#Updates
NoNeg1	2	3	25	34	12	233
NoNeg2	3	4	45	88	46	608
NoNeg3	4	5	66	104	42	655
NoNeg4	6	7	69	149	85	1107
Neg1	2	3	22	24	4	139
Neg2	3	4	59	106	50	779
Neg3	4	5	41	57	20	423
Neg4	5	6	74	114	45	834

Table 3.5. Statistics for each genome developing networks for each logic program using fixed event sequences. #Conns refers to the number of connections, #Add and #Delete are the total number of connections added and removed, and #nUpdates is the total number of weight updates. All test questions are answered correctly in each case. Each logic program and developed network is a different size, despite the fact that the same fixed-size genome is used in each case. This shows that the genome is scalable.

Program	#Connections	#Add	#Delete	#Updates
G1	197.5	194	0	1454.38
G2	117.6	205	90.5	1282.63
G3	78.25	75	0	663.5
G4	57.5	100	46	697.63

Table 3.6. Average statistics from table 3.5 for each genome



(a) G1 (b) G2

(c) G3 (d) G4

Figure 3.12. Time taken to develop networks of different sizes using each genome. Although the genomes are scalable in the sense that each genome can develop networks of different sizes despite being of a fixed size, they are not scalable in terms of evaluation time. The time taken to evaluate each network as it develops is O(n), roughly linearly proportional to the number of connections in a network.

with or without negated predicates. A fixed event sequence was used in each case to ensure that all genomes were tested on the same data. Each sequence was repeated until a total of 500 event observations had been made, with a window of synchrony of two observations. Developed networks were presented with the corresponding B question set to demonstrate each network's ability to correctly assert all conclusions. Event sequences and question sets may be found in appendix A. In order to overcome the conflicting relations problem for Neg1-4, *hInc* (page 76) was set to 1.25 and *hDec* was set to 1 to enforce a bias towards strengthening connection weights.

In each case all test questions were answered correctly. Graphs of the change in fitness (measured as the number of correct answers to questions) as the networks learn and develop can be found in fig. 3.10 on page 83, and tables 3.5 and 3.6 on page 84 present the statistics of the network for each logic program developed using each of the genomes. Statistics recorded for each network include the final number of connections formed, the total number of connections added and removed during development, and the number of weight updates performed during learning. Fig. 3.11 on page 83 shows the final number of connections and number of weight updates yielded by each genome when developing networks for all logic programs. All networks developed contain a different number of connections despite the fact that they were all developed using the same fixed size genome. This demonstrates that like other indirect encodings in the literature, the genome is scalable in the sense

that the size of the genotype is independent of that of the phenotype. However, note that the genomes are not scalable with respect to the time taken (in seconds) to evaluate each network as it develops. The time taken to evaluate a network as it develops is O(n), in that the relationship between the number of connections in a network and its evaluation time in seconds is roughly linear (fig. 3.12).

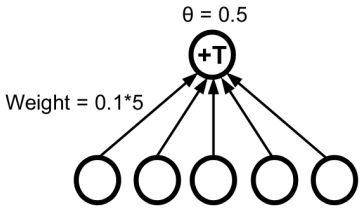
Although these experiments only demonstrate scalability to logic programs in which predicates have no more than three arguments[5], it is argued that the genome would scale to relations between larger predicates because the genome represents a connection between two arbitrary role (argument) nodes that can be constructed as often as is required, irrespective of how many arguments a predicate has. A stronger defence of this argument is presented later in section 3.5.

G1 developed the largest network, with the greatest number of connections and weight updates. This follows from the fact that the genome represents the bare minimum required for SHRUTI's learning algorithm to work: a network in which all predicate nodes of the same function are connected with no attempt to reduce the number of connections. All other genomes make some attempt to reduce the number of connections and the results show that they are successful. G4, which restricts the formation of connections to active neurons of the same function and removes redundant connections, appears to be the most successful at reducing the final number of connections. However, the number of additions, and to a lesser extent the number of weight updates, is slightly higher than that of G3, which only restricts connections to active neurons of the same function but does not remove redundant connections. This is owed to the fact that when a connection between two neurons is removed, it is likely that it will eventually be reconstructed, and as a result the genome actually does more work by constructing a greater number of connections overall. In conclusion, it is more efficient to prevent the development of superfluous connections than it is to prune them. G3 is therefore the most efficient genome.

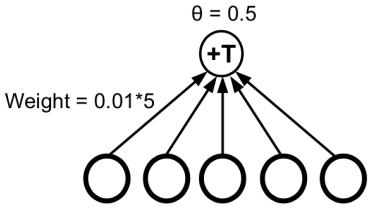
For each genome, figure 3.10 shows the change in fitness, measured as the number of correct answers, for NoNeg4 and Neg4 over fixed event sequences. Maximum fitness is achieved in each case.

However, the fitness of G2 oscillates and doesn't actually rest at maximum fitness. This is caused by frequent activation of certain collectors when the sum weight of weak inputs is equivalent to one strong, above threshold input. For example, in NoNeg4, +T receives inputs from more than five neurons, with each input having a weight of 0.1 (figure 3.13a). The sum weight of these inputs is 0.5, enough to cross the threshold and activate +T when it would otherwise remain inactive. This happens early in development for both G1 and G2, but connections are eventually weakened enough to unlearn the disruption (figure 3.13b). This solves the problem for G1, but in G2 these connections are eventually pruned (figure 3.13c) and later reconstructed (figure 3.13d). Because these connections are reconstructed with their original weight values of 0.1, +T reactivates when it shouldn't and fitness drops again. This occurs in G2 each time these connections are removed and reconstructed, hence the oscillation in fitness. Modifying the genome so that the initial

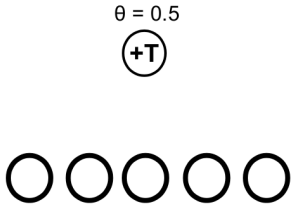
[5]The reader is referred to page 71 for a defence as to why a limit of three arguments was set.



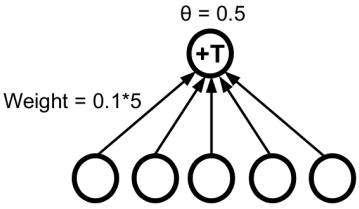
(a) Connections created, +T active.



(b) Connections weakened, +T inactive.



(c) Connections deleted, +T inactive



(d) Connections recreated, +T active.

Figure 3.13. Problem developing connections with genome G2, which prunes weak connections. A set of inputs for which individual weights are initially 0.1 but the sum weight crosses the activation threshold (0.5) is repeatedly removed and reproduced by the genome, resulting in the output periodically becoming activated for the wrong questions.

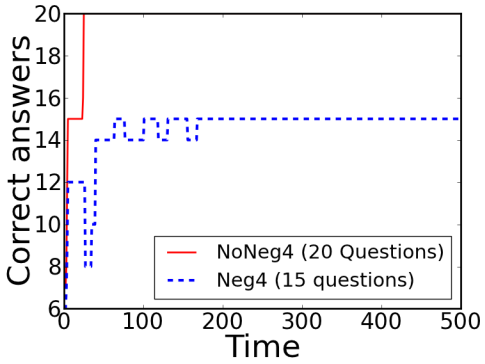
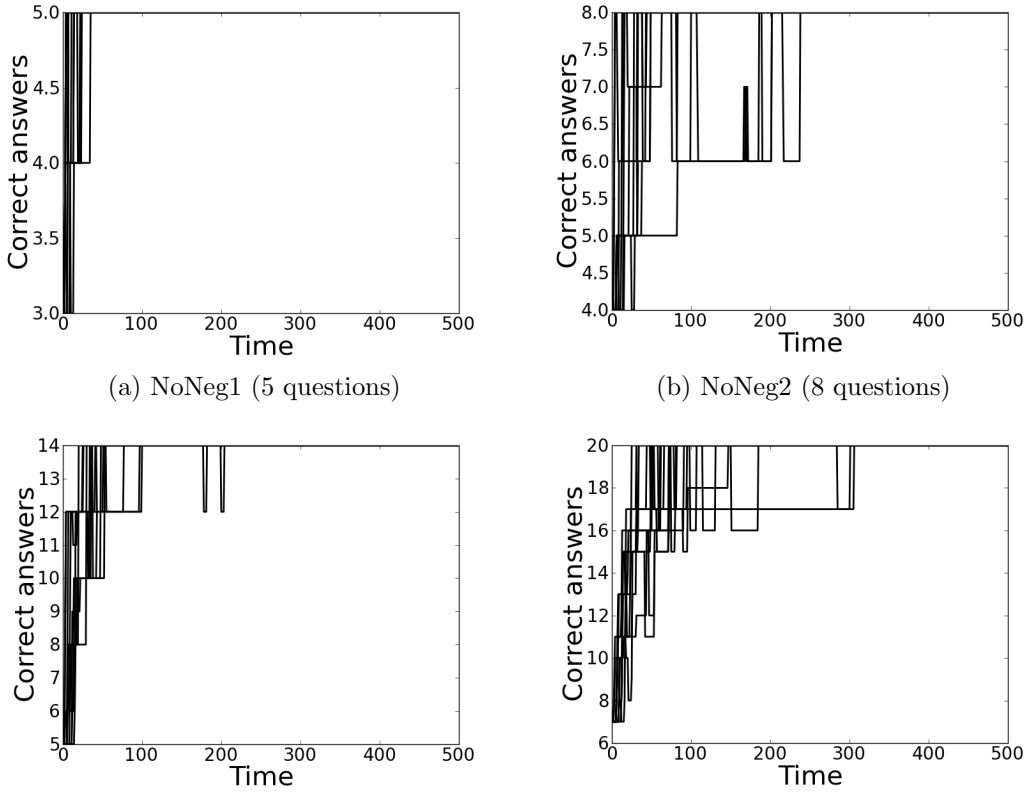


Figure 3.14. Fitness of networks developed using G2 with initial weight value reduced.

weight value of a new connection is 0.05 instead of 0.1 means that +T only receives a below threshold activation of 0.25 and development can occur without any oscillation in fitness (figure 3.14). Nonetheless, regardless of how weak the input is, the problem could occur in any network where a sufficient number of weak inputs is strong enough to activate a neuron at the wrong time. This problem presents another reason why it is better to prevent the development of superfluous connections than it is to prune them.

All genomes were also successful at developing zero-error networks (networks that answer all questions correctly) for probabilistic event sequences for the NoNeg logic programs[6]. Figure 3.15 shows the number of questions answered correctly over time for NoNeg networks developed using G4. Although the same results for other genomes are excluded for brevity, the performance of G1 is exactly the same as that shown earlier in figure 3.2, since G1 produces a fully interconnected network. Because relations cannot be learned using the probabilistic sets for the Neg data as argued in section 3.1.2, the genomes could not produce zero-error networks for Neg programs the same reason.

[6] Although for G2 this was only true when modifying the initial weight of a new connection to 0.05, as when trained on fixed sequences (fig. 3.14)



(a) NoNeg1 (5 questions)

(b) NoNeg2 (8 questions)

(c) NoNeg3 (14 questions)

(d) NoNeg4 (20 questions)

Figure 3.15. Fitness (measured as the number of correct answers) of developing networks trained on probabilistic event sequences. 10 trials are performed in each case, generating a new event sequence each time according to the probabilities listed in appendix A.

3.2.4. Discussion

The genome required to create connections between nodes in a SHRUTI network such that zero-error is achieved is very small. The bare minimum required is for two neurons to be of the same function, requiring a bare minimum of one condition (P_INPUT.function = SELF.function) and one action (ADD). Creating a smaller, more efficient network requires two extra conditions that restrict the development of connections to active neurons (SELF.activity >0.5, P_INPUT.activity >0.5).

As discussed in section 2.1.6, synaptic activity is known to influence gene expression in such a way that leads to neural development, including the maturation and removal of synapses. Activity-dependent gene expression may not occur as directly and immediately in biology as it does in this model, but this model is intended only to be an abstraction just as SHRUTI itself is an abstraction. What one may question is the fact that these activity-dependent conditions that restrict development parallel an aspect of the learning algorithm: that of strengthening connections between co-active nodes. One would expect neural development to support learning but not to copy it, otherwise learning is redundant. Furthermore, although connections between neurons do form in post-natal development, many neural structures are produced during the pre-natal stage before any learning can even take place. A biologically plausible genome would therefore perform in a similar manner by first performing some development independent of learning rather than being

influenced by learning from the very beginning. However given the SHRUTI structure, it is difficult to see how the size of a network can be restricted during development without activity-dependent gene expression that mimics the learning algorithm.

In order to ensure that target relations can be learned using the representation method used in SHRUTI, there are three possibilities if development is not to be influenced by event observations that lead to neural activation:

1. Structures for all possible relations already exist in the network or eventually appear at some point in time determined by the genome alone. However, such a fully interconnected network is exactly what we are trying to avoid for the sake of biological plausibility and efficiency.

2. Only structures for the target relations exist to begin with. However this would suggest that all knowledge is innate and that learning is redundant. Although some human knowledge may be innate, surely the majority of it is not.

3. Structures for the target relations plus others pre-exist. However this still requires that the target knowledge is innate.

If the environment does affect neural development, it must be done through sensory perception, either directly or indirectly. The only interaction a SHRUTI network has with the environment is through the activation of its predicate nodes. Three possibilities for ensuring that target relations can be learned exist in this case:

1. Structures for all possible relations that a predicate may participate in develop when that predicate is observed and its nodes are activated.

2. Structures for the target relations develop when the antecedent and consequent are observed to coincide and their nodes are coactivated, as in genomes G3 and G4.

3. Target structures plus others pre-exist and weak connections not supported by evidence are pruned.

In all three cases, some element of the learning algorithm is evident in the developmental process, since the learning algorithm is also dependent on the activity or inactivity of neurons.

The current SHRUTI representation is localist in that each relation is represented by a specific structure reserved for that relation. Therefore in order for a particular relational structure to be learned by Hebbian or even recruitment learning, it must exist to be learned in the first place. The above paragraphs argue that given the current SHRUTI model, this can only occur though pre-existence or observation. This may not be the case if a different representational structure were to be found. A further discussion on this is presented at the end of the chapter.

Regardless of what leads to a structure's construction, this work shows that relational structures can be represented by indirect encodings. Therefore the claim of SHRUTI's developers that the prerequisite structure required for SHRUTI to learn relations can be

realised through a biologically plausible model of development already carries some weight. Furthermore, the representation is scalable in the sense that the same fixed size genome can be used to develop networks of different sizes. Particularly biologically plausible traits include the use of indirect encoding to produce repeated substructures [22, 103, 116], gene regulation including the existence of introns and exons [116], and activity-dependent development [33]. Of course, the model is still an abstraction of true biological development and biological plausibility could still be improved.

The next step was to attempt the production of similar SHRUTI genomes through an evolutionary process. Owing to the simplicity of these genomes, it was expected that similar genomes would be easily found in an evolutionary search. Nonetheless, evolving SHRUTI genomes was still worthwhile in order to find any possible alternatives to the genomes presented above, to demonstrate the concept of evolvable relational structures before attempting to evolve genomes for more complex structures, and to observe any potential difficulties that may be encountered when doing so. Furthermore, if alternative methods of restricting development that do not depend on pre-existence or coactivation do exist after all, it was expected that they would be found by the evolutionary search.

3.3. Method for evolving SHRUTI genomes

Now that a developmental genome had been produced, the next task was to observe whether such genomes could be found through evolution, with the additional aim of finding alternatives or improvements to the genomes in section 3.2, if they exist. The work in this section and the remainder of this chapter extends work originally presented by the author in [113, 114].

The multi-objective evolutionary search NSGA-II [29] was chosen because there were two objectives: to reduce error in the answers produced by developed networks and to reduce the number of connections necessary to do so. The parameters used to evolve networks with NSGA-II are described in section 3.3.1, and the variation methods used are described in section 3.3.2. Training and test data are explained in section 3.3.3 and section 3.3.4 introduces the error function.

Both objectives, error and number of connections, could be measured indirectly as the area beneath the graph of error over time (*e-area*) and number of weight updates performed during learning, respectively. Therefore before the main evolutionary experiments began, the advantages of measuring the objectives in this way were explored. The method for these preliminary experiments is described in section 3.3.5.

To ascertain whether or not evolution could discover improvements or alternatives to the SHRUTI model, it was necessary to sample evolved genomes in order to examine their structure and that of the networks they develop. Section 3.3.6 explains the sampling method used when selecting genomes from the final populations of evolutionary trials.

Before continuing, it is important to note that the evolutionary process and the muta-

tion and crossover methods used are only biologically plausible in the sense that they are abstractions of biological processes. This is fitting with the definition of biological plausibility in section 2.1.7, and further biological plausibility is a point for future work.

3.3.1. Parameters

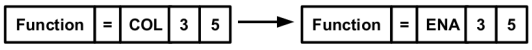
The genomes were evolved over 500 generations with a population size of 100, and 50 trials were performed for each logic program. Parents for variation were selected using tournament selection with a tournament size of 2. The crossover and mutation rates were 0.9 and 0.1 respectively. Genomes had a fixed size of twelve conditions or actions, as this is more than sufficient to represent genomes G1-4 from section 3.2.

3.3.2. Variation

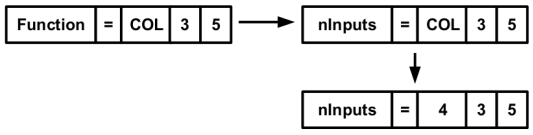
As explained in section 2.4, variation is the process through which genomes pass on their genes to the next generation. Variation operations were performed until the size of the offspring population was equal to that of the parent population. Two operations were possible: the crossover of two parents followed by the mutation of each child, or the mutation of one parent. Which of these was performed for each operation was determined by the crossover rate. In either case, the mutation rate determined the probability of a gene's mutation. Pseudo-code for both variation operations can be found in appendix C.

Genes in introns and exons were candidates for mutation. When a gene was selected for mutation, a new value was chosen according to a uniform distribution (fig. 3.16a). In the event that a gene was mutated in such a way that other genes within the condition or action were now incompatible, the incompatible genes were also mutated. For example. in fig. 3.16b, the attribute is mutated from 'function' to 'nInputs' (number of inputs). Because the number of inputs requires a numerical argument, 'COL' is an incompatible argument and is therefore mutated into a random integer.

Crossover was performed by producing copies of two parent genomes and swapping intron and exon nodes between them. Consider two parents P_1 and P_2, and a set of nodes N_i from each parent. Each N_i is composed of a sub-tree of exons tr_i and a set of introns I_i such that the lengths of N_1 and N_2 are equal. To produce the first offspring, P_1 is copied

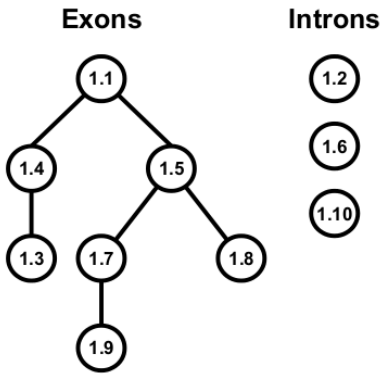


(a) Standard mutation: argument 'COL' (collector) is mutated into 'ENA' (enabler)

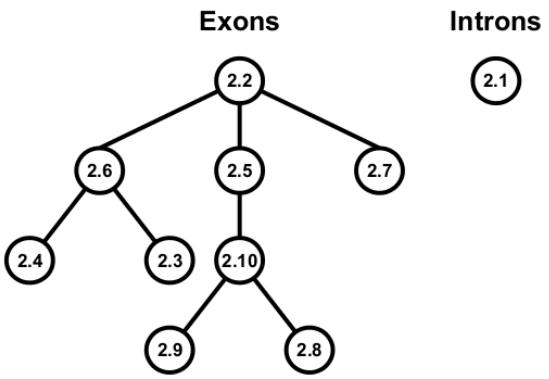


(b) Attribute 'Function' is mutated into 'nInputs', which is incompatible with argument 'COL', so the argument is also mutated.

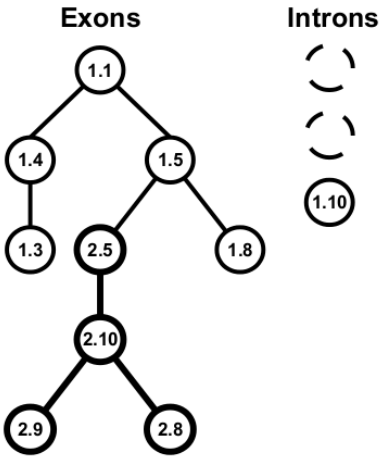
Figure 3.16. Mutation. Each diagram represents the mutation of a condition, and each square in a condition represents a gene. Each gene has a probability of mutation equal to the mutation rate. If a gene is mutated that another gene is dependent on, that second gene is also mutated (b).



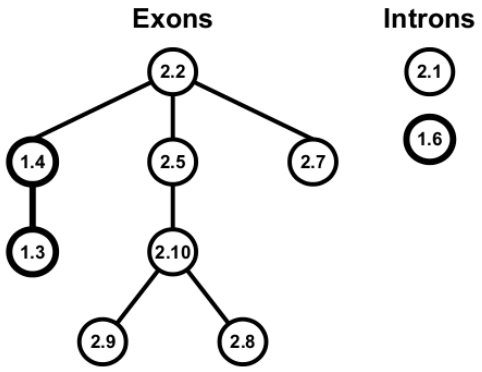
(a) Parent 1



(b) Parent 2



(c) Child 1: Copied from parent 1. The sub-tree previously at exon 1.7, and introns 1.2 and 1.6, are replaced by a copy of the sub-tree at exon 2.5 from parent 2.



(d) Child 2: Copied from parent 2. The sub-tree previously at exon 2.6 is replaced by copies of nodes from parent 1: intron 1.6 and the sub-tree at exon 1.4.

Figure 3.17. Crossover. To produce child n, a copy of parent n is first made. From child n, a tree of exons and a set of introns are randomly chosen to be replaced by randomly chosen copies of an exon tree and set of introns from the other parent. Nodes are always chosen in such a way that the total number of nodes in a genome is preserved after crossover.

to produce C_1. N_1 is removed from C_1 and replaced with a copy of N_2, with the root of tr_2 placed in the original index of the root of tr_1. For example, to produce child 1 in fig. 3.17, $tr_1 = \{1.7, 1.9\}$, $I_1 = \{1.2, 1.6\}$, $tr_2 = \{2.5, 2.8, 2.9, 2.10\}$ and $I_2 = \{\}$. Therefore $N_1 = \{1.2, 1.6, 1.7, 1.9\}$, $N_2 = \{2.5, 2.8, 2.9, 2.10\}$ and $length(N_1) = length(N_2) = 4$. In fig. 3.17c, parent 1 is copied to create child 1, tr_1 (starting at exon 1.7) and introns 1.2 and 1.6 (I_1) are removed from child 1 and replaced with a copy of tr_2 (starting at exon 2.5) and the empty set I_2. To produce the second offspring, P_2 is copied to produce C_2, new sets N_1 and N_2 are generated and the process repeats except that the new N_2 is now replaced by the new N_1. For example, when generating the second child in fig. 3.17, $tr_1 = \{1.3, 1.4\}$, $I_1 = \{1.6\}$, $tr_2 = \{2.3, 2.4, 2.6\}$ and $I_2 = \{\}$. The new N_2 (2.3, 2.4, 2.6) is then replaced by the new N_1 (1.3, 1.4, 1.6) as shown in fig 3.17d. In the production of all offspring, all sets were chosen according to a uniform distribution, but the ordering of node types specified in section 3.2.2 had to be enforced. For example, if $root(tr_i)$ denotes the root of a tree, then when N_1 is replaced by N_2, tr_1 and tr_2 must be chosen in such a way that $type(root(tr_1)) \preceq type(root(tr_2))$.

3.3.3. Training and test data

Genomes were evolved for all eight logic programs listed above in section 3.1. The fitness of each genome was assessed by using it to develop a network based on event sequences and testing the developed network on a set of training questions. Fixed event sequences listed in appendix A were used for training and developing networks to ensure that each member of the population was trained on the same data. Event sequences were repeated until 60 observations[7] had been made, to bias the evolutionary search towards genomes that develop networks that could learn to yield minimum error within this time. The window of synchrony was once again set to two observations.

Developed networks were presented a set of questions in order to calculate the error, using the A question set corresponding to the logic program represented by each network (appendix A). The error function is described in section 3.3.4. The A set was chosen because it contains one question for the consequent of each relation, and it was hypothesised that if a network developed to answer one question correctly for a particular relation, it would be able to do so for all questions dependent on that relation, since all questions dependent on a particular relation are answered using the same relational structure. Furthermore, a question for which the expected answer is 'unknown' is included for every predicate in order to avoid convergence towards networks which always answer 'true' or 'false' without considering argument bindings.

Final populations for all logic programs were tested on longer event sequences, as opposed to the first 60 observations on which they were trained, in order to observe whether or not zero-error genomes maintain their efficacy over longer periods. These longer periods were produced by repeating event sequences three times. These tests were performed first on the same question sets used in evolution, and then again on a set of questions enumerating all possible questions the network can be asked, in order to test the hypothesis that evolving on one question per relation allows generalisation to larger question sets.

Genomes from the final population evolved for each logic program were then tested further by developing networks for all other logic programs using the event sequences and test questions for those programs. For example, genomes evolved to represent NoNeg4 were tested by developing networks for NoNeg1-3 and Neg1-4. These tests were performed to test the hypothesis that genomes generalise well to other logic programs because they are evolved to represent the very concept of a logical relation, and not the representation of any particular logic program as a whole.

Finally, the evolved genomes were then tested on probabilistic event sequences supporting the same logic programs in order to test how adaptable genomes evolved on fixed sequences were to noisy, probabilistic event sequences. 500 observations were generated according to the probabilities specified in appendix A. Each genome was tested on 10 random seeds, taking the average error yielded from each. Because this increased execution time tenfold, samples were taken from each trial in order to reduce overall computation time. Up to five zero-error genomes were chosen at random from the Pareto front of each trial if five

[7] The reader is reminded that time steps in which no events occur are nonetheless regarded as observations.

such genomes existed, otherwise only the few that did exist were chosen. The enumerated question sets were used, and genomes chosen for each logic program were tested on their own event sequences and those for other programs.

3.3.4. Evaluating error

Error was measured as the difference between maximum accuracy and the accuracy obtained, where accuracy is based on how many correct answers a network gives to test questions. However, instead of counting the number of correct answers, accuracy was based on the number of correct collector activations. Each answer to a question is read as the state of the positive and negative collector [+, -], and therefore takes one of four values: [1,0] (true), [0,1] (false), [0,0] (unknown) and [1,1] (contradiction).

The complete set of answers is represented as a matrix a, in which each row a_i represents one answer and contains an element for each collector activation. a is compared against a target matrix t and each row a_i in the answer matrix is assigned a score. Total accuracy is then measured as the sum of these scores as in equation 3.2. Originally, a simple scoring function in which a score of 1 is assigned to each correct collector activation was considered (equation 3.3). However this function meant that by always answering [0, 0] (unknown) for each question, a high percentage accuracy could be obtained since all correct answers contain at least one inactive collector state, as shown in table 3.7 on the next page. The scoring function was modified to overcome this problem (equation 3.4). Networks which always answer [0,0] are penalised to receive a total score of 0, and more weight is given to correct 'True' or 'False' questions by assigning a score of 3, not 2, when both collector values are correct. As shown in table 3.7, networks only answering [0,0] are now assigned the absolute minimum accuracy, and the accuracy of networks answering all questions correctly is even greater than before.

$$Accuracy(a) = \sum_{i=0}^{n} S(a_i) \tag{3.2}$$

$$S_1(a_i) = \sum_{j=0}^{m} (1 - |t_{i,j} - a_{i,j}|) \tag{3.3}$$

$$S_2(a_i) = \begin{cases} 0 & \text{if} \sum_{i=0}^{n} \sum_{j=0}^{m} a_{i,j} = 0 \\ 3 & \text{if} \sum_{j=0}^{m} t_{i,j} = 1 \text{ and } \sum_{j=0}^{m} |t_{i,j} - a_{i,j}| = 0 \\ S_1(a_i) & \text{otherwise} \end{cases} \tag{3.4}$$

3.3.5. Alternative objectives

The two objectives to be minimised were the error yielded by a network and the final number of connections in the network. However, the option of measuring both of these

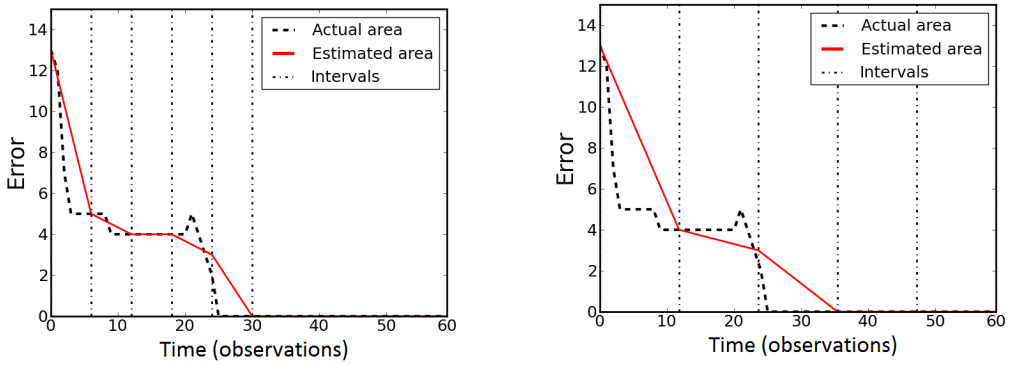
Expected answer		Score 1 (S_1)			Score 2 (S_2)		
+	-	All 0,0	All 1,1	All correct	All 0,0	All 1,1	All correct
1	0	1	1	2	0	1	3
0	1	1	1	2	0	1	3
0	0	2	0	2	0	0	2
0	0	2	0	2	0	0	2
		6	**2**	**8**	**0**	**2**	**10**

Table 3.7. A comparison of the outputs of two different scoring functions. Numbers in bold denote total accuracies. Using the first scoring function, networks can achieve a high accuracy by simply answering [0,0] (unknown) for every question. This is not the case with the second scoring function.

objectives indirectly was also considered.

The alternative to the first objective of minimising error at the end of development was to minimise the area beneath the graph of error over time (referred to as '*e-area*' henceforth), which is representative of the error of a network as it develops. Measuring the area beneath the error-time graph provides information about the learning speed of a network and what error values are yielded during development. Furthermore, the e-area provides a more granular means of measuring performance which leads to increased diversity in phenotypic space. Measuring error alone yields a smaller and more discrete range of values which results in lower phenotypic diversity. However, the disadvantage of this alternative approach is that measuring the e-area involves measuring error over numerous intervals instead of just once at the end of development. Execution time increases with the number of intervals at which error is measured, and can become particularly expensive for large question sets since one pass of the network must be performed for each question.

As described above, networks are only trained on 60 event observations to encourage early learning. The means by which area was estimated was chosen with a similar goal in mind. Fig. 3.18a shows the error-time graph for a network that learns quickly. Error is measured five times over the first 30 observations and once again at the 60th observation



(a) Area estimation 1 (b) Area estimation 2

Figure 3.18. Estimating area for developing SHRUTI networks by measuring error at intervals. The maximum error in this example is 15. In the first approach, error is measured over five intervals for the first thirty observations and once again at the sixtieth to assert that the network maintains its efficacy. The second approach simply measures error at five intervals over all sixty observations, but the first approach is preferred as it encourages the discovery of networks that learn quickly.

Objective	Alternative objective
Minimise final error	Minimise error under error-time curve (e-area)
Minimise final number of connections	Minimise total number of weight updates

Table 3.8. For both objectives (error and number of connections), there exists an alternative measurement as shown in this table.

for validation to ensure that error remains at zero. This provided a better estimation of the actual error-time curve than the alternative of spreading the five intervals over all 60 observations (Fig. 3.18b) and encourages the evolutionary algorithm to find networks that learn all relations within the first 30 observations.

For the second objective, the alternative to measuring the final number of connections in the network was to measure the total number of weight updates performed during development. Once again, the two statistics are related, because a greater number of connections results in a greater number of connection weights and therefore a greater number of weight updates. Minimising the number of weight updates will therefore constrain the number of connections and in addition minimise the work of the learning algorithm. Although measuring the number of weight updates was more computationally expensive than measuring the final number of connections alone, the difference was negligible.

Each objective and its alternative is summarised in table 3.8. In order to choose a suitable pair of objectives for evolving networks for all logic programs, each possible combination was tested on the evolution of networks for the NoNeg4 program alone (section 3.4.1). When minimising each pair of objectives, values for each objective's alternative were also recorded to observe how they were affected. For example, when the e-area and number of weight updates were selected as objectives, final error and number of connections were also recorded to observe whether or not these values were constrained by the minimisation of their alternatives. Once a suitable pair of objectives had been chosen based on these results, networks for all logic programs were then evolved using the same parameters and the chosen objectives (section 3.4.2).

3.3.6. Sampling evolved genomes

In addition to exploring whether or not SHRUTI genomes could be evolved, secondary aims were to explore whether or not any improvements or alternatives to the SHRUTI model could be discovered through evolution. To do this, it was necessary to examine the structure of evolved zero-error genomes and the networks they developed. However, it was impractical to examine every zero-error genome evolved, and therefore genomes were sampled by selecting one zero-error genome for each value of e-area on the final Pareto front for every 10th trial for each test scenario.

3.4. Evolution of networks

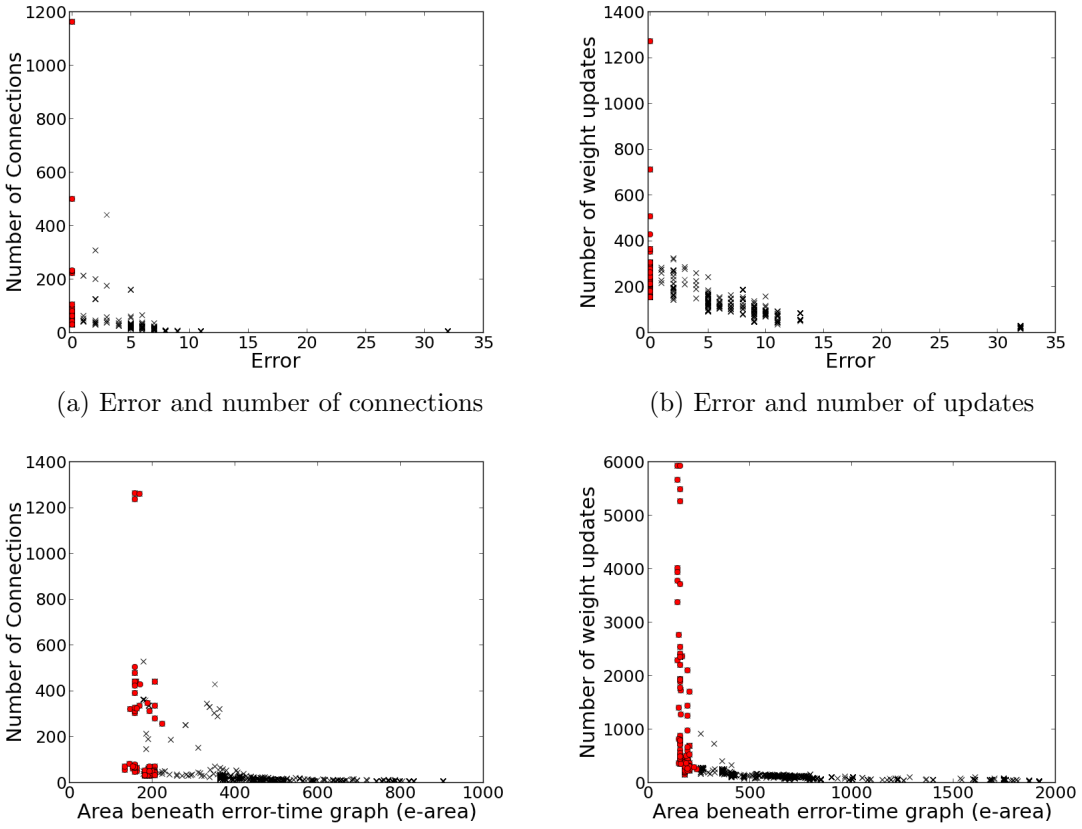
This section presents the results of evolving genomes for developing SHRUTI network representations of the logic programs listed in section 3.1. The preliminary analysis on possible objective pairs is given in section 3.4.1. Section 3.4.2 presents the results of using the chosen objective pair to evolve genomes for all logic programs and section 3.4.3 presents the results of using the evolved genomes to develop networks for test data.

3.4.1. Selection of objectives

Before evolving genomes for all logic programs, a choice of objectives to minimise had to be made. The choices to be made were between error and e-area, and between the final number of connections and the total number of weight updates. The preliminary

Objective pair	Trials with zero-error networks (out of 50)
Error and number of connections	47
Error and number of updates	49
Area and number of connections	41
Area and number of updates	48

Table 3.9. The number of zero-error networks produced from experiments minimising each pair of objectives.



(a) Error and number of connections

(b) Error and number of updates

(c) Area and number of connections

(d) Area and number of updates

Figure 3.19. Pareto fronts produced for different objective pairs when evolving SHRUTI genomes. Points marked with a red dot indicate networks that developed to yield zero-error.

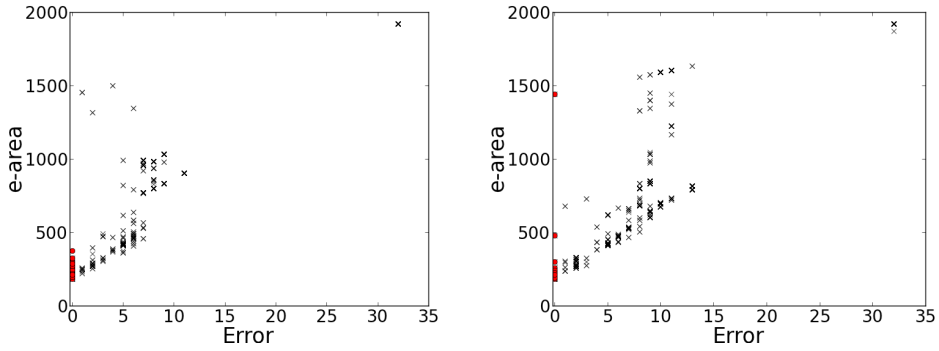
experiments described in section 3.3.5 were performed on the NoNeg4 data set in order to observe the effect that each possible combination of objectives had on other statistics in the evolved networks and thereby choose a suitable pair of objectives for evolving genomes for all other logic programs. For each pair of objectives, table 3.9 shows the number of trials in which networks which yielded an error of zero were found and fig. 3.19 shows the Pareto fronts obtained for all 50 trials. Networks which yielded an error of zero will be referred to as *zero-error networks* henceforth, and are indicated by red dots in the plots.

The Pareto fronts represent a trade-off between error/e-area and the number of connections/weight updates. A small number of connections means that the learning algorithm has to do less work, but if too small, the network will be unable to yield zero-error. A greater number of connections does not guarantee a zero-error network because connections need to be arranged correctly, but even when properly arranged too many connections results in a network that uses more resources than necessary. In summary, an ideal network is one in which the number of connections/weight updates is the smallest it can possibly be while still enabling the network to learn to yield zero-error as early as possible. Such points are those both marked in red and sitting in the knee-point of each Pareto front.

The worst results in terms of the number of trials that discovered zero-error networks were obtained when both area and number of connections were chosen as objectives, but there is little difference between the performance of other combinations of objectives (table 3.9). However Pareto curves do differ in that they are more distributed when minimising e-area instead of error, for reasons which will be explained below.

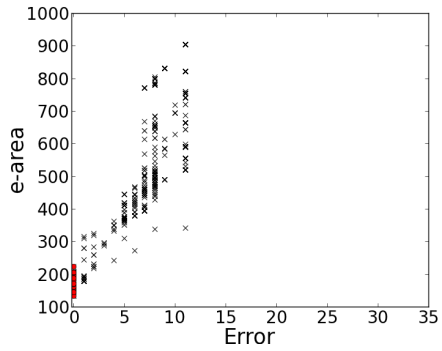
Error versus e-area

Fig. 3.20 on the next page shows the error and e-area yielded by each network in the Pareto front of each trial for each objective pair. In each experiment there is some correlation between error and area, suggesting that one of these statistics provides a fairly accurate indication of the other. This correlation exists because of the very fact that e-area is a measure of error over time. If error is greater at the end of development, the final e-area will proportionally greater. The e-area and the number of connections form the objective pair that underperforms compared to the others (table 3.9), with only 41 trials yielding zero-error networks. Furthermore, one of the goals of using e-area as an objective was to encourage early learning, but reducing the number of weight updates during learning has the same effect. For these reasons it would appear that error is a slightly better choice of objective, especially since measuring area is more computationally expensive. However, the e-area provides one piece of information that no other objective does: what error values have been yielded *during* development. Two networks may yield the same error, but if one of them yields a smaller e-area then it temporarily yields a smaller error earlier in the learning process (fig. 3.22, page 100). Of the two, the network temporarily yielding the smaller error is closer to the ideal zero-error network in terms of phenotypic behaviour and therefore of more interest to the evolutionary search. Although this difference did not appear to significantly affect the results for evolving networks for NoNeg4, it may

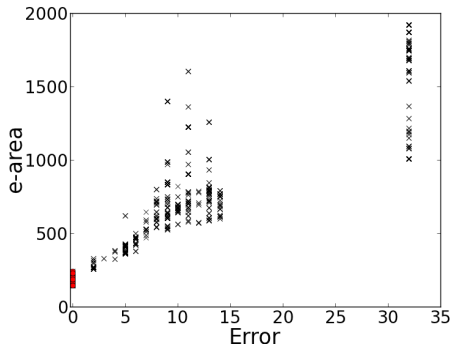


(a) Error and number of connections

(b) Error and number of updates

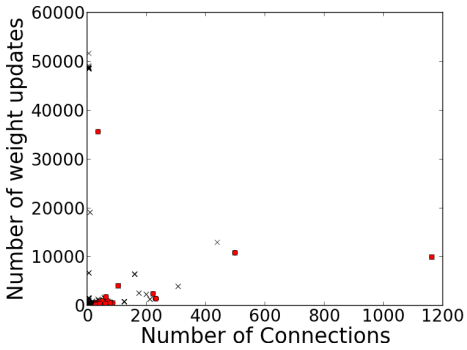


(c) Area and number of connections

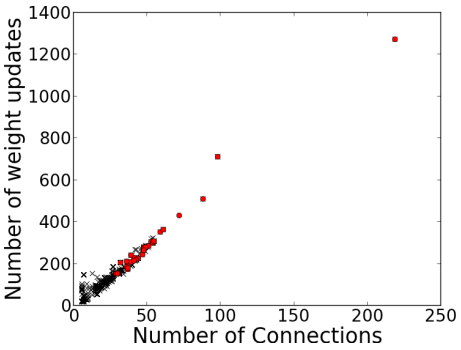


(d) Area and number of updates

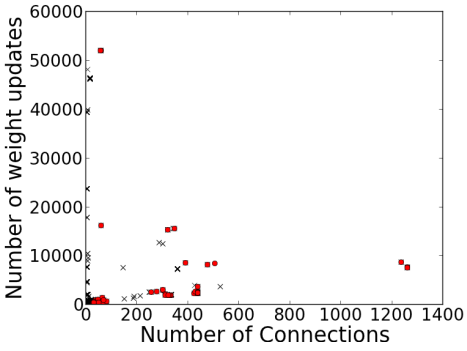
Figure 3.20. The correlation between the error and the e-area for each pair of objectives. In each case, the two statistics are found to correlate, especially for smaller error values.



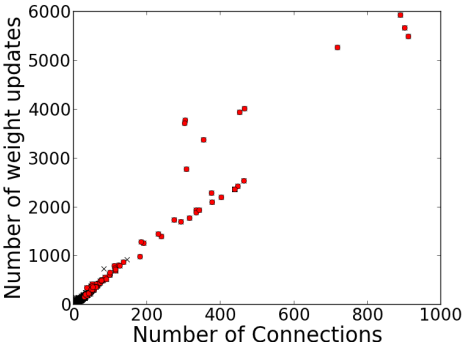
(a) Error and number of connections



(b) Error and number of updates

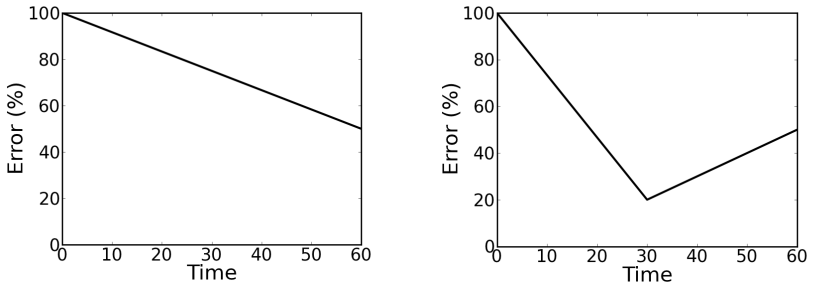


(c) Area and number of connections



(d) Area and number of updates

Figure 3.21. Correlation between the number of connections and the number of weight updates for each pair of objectives. The two statistics only correlate when the latter is chosen as an objective.



(a) Larger e-area (b) Smaller e-area

Figure 3.22. Two networks yield the same error (50%) after learning, but may be distinguished by a smaller e-area that indicates that the second network temporarily yields a lower error.

do for other logic programs, and therefore e-area was selected over error as an objective for further experiments. Note also that because e-area takes into account error values at different points in time, minimising e-area instead of error at the end of development alone yields a broader range of objective values and therefore a greater distribution of points in the Pareto front.

Final number of connections versus total number of weight updates

Fig. 3.21 shows the final number of connections and the total number of weight updates for each network in the Pareto front of each trial for each objective pair. The number of updates and the number of connections only correlate when the number of updates, and not the number of connections, is selected as the second objective. Measurements of both values are taken at the end of the developmental process. The final number of connections does not reflect the number of connections that have ever existed during development and therefore how many weight updates have been made because some connections may have been removed during development. However, the number of connections that have ever existed, and therefore also the total number of weight updates performed on them, must always be greater than or equal to the final number of connections. Thus, minimising the final number of connections cannot constrain the number of updates but the final number of connections can be constrained by minimising the number of updates. In conclusion, the number of updates is preferred over the final number of connections as an evolutionary objective because minimising the former has the effect of constraining both.

In summary, although there was little difference in performance between each pair of objectives, the e-area and the number of weight updates were chosen as the objectives to minimise in further experiments in order to account for the following properties of evolved networks when selecting genomes for crossover and mutation:

- Final error value
- Error values encountered during development
- Final number of connections
- Learning speed

3.4.2. Performance on training data

Using the selected objectives and the methodology described in section 3.3, networks for the remaining logic programs presented in sections 3.1.1 and 3.1.2 were evolved. Figure 3.24 shows samples obtained from 50 trials for each logic program when minimising e-area and the number of weight updates. Networks which yielded zero error are indicated by a red dot. Table 3.10 shows for each logic program the number of trials that developed zero-error networks and the total number of zero-error networks across those trials. Most trials were successful in producing zero-error networks for each logic program.

Fig. 3.23 shows an example of Pareto front development for NoNeg4, excluding the initial population. Zero-error networks tend to be discovered within the first 100 generations. This supports the hypothesis made at the end of section 3.2 that zero-error networks would be easy for evolution to discover. However, the number of weight updates for these earlier networks was relatively high compared to those in later generations. Therefore although finding zero-error networks appears to be a relatively simple task for the algorithm, minimising the number of weight updates and therefore improving the learning speed required to do so is not, as this tends to take place over a longer period of evolutionary time.

Genomes produced by the evolutionary search were examined by sampling one genome for each value of e-area on the final Pareto front for every 10th trial for each test scenario. Surprisingly, most of the genomes sampled fell into in one of three groups which can be

Logic Program	NoNeg1	NoNeg2	NoNeg3	NoNeg4	Neg1	Neg2	Neg3	Neg4
Trials with zero-error networks	50	49	49	49	50	50	49	49
Number of zero-error networks	581	363	255	469	282	219	194	332

Table 3.10. Statistics of trials for each logic program. In total, 2695 zero-error networks were found.

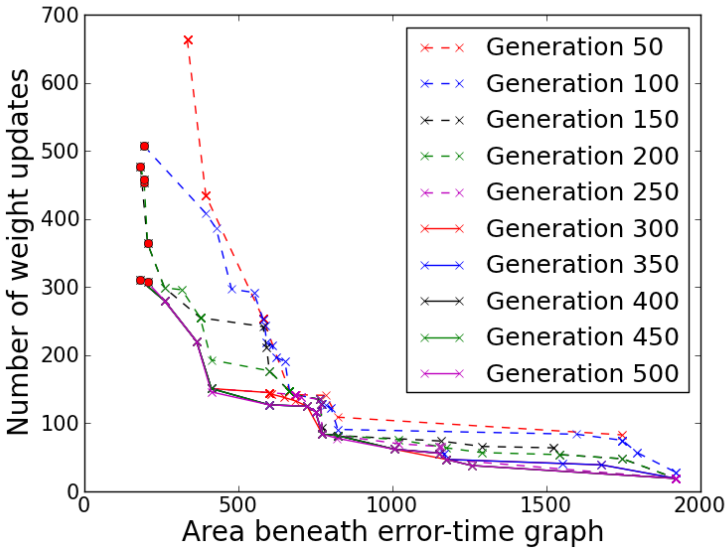
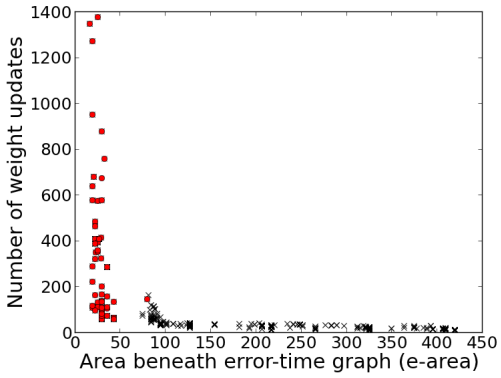
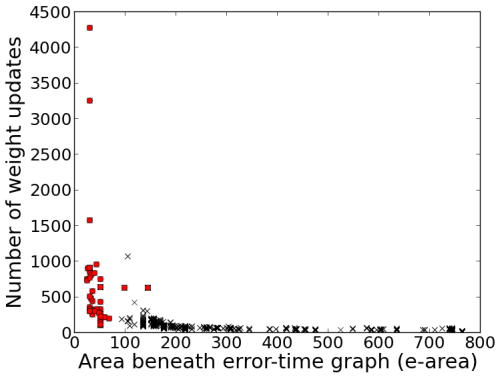


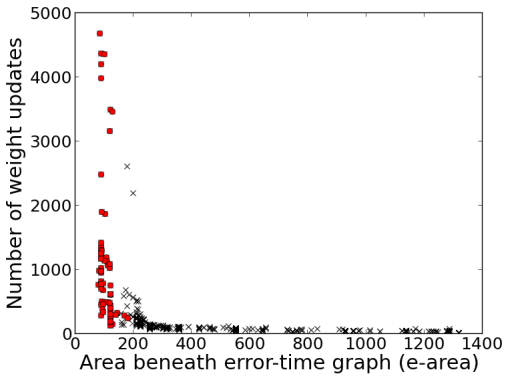
Figure 3.23. Pareto fronts at different stages of evolution.



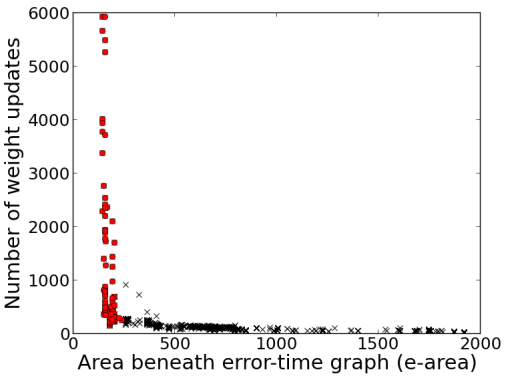
(a) NoNeg1



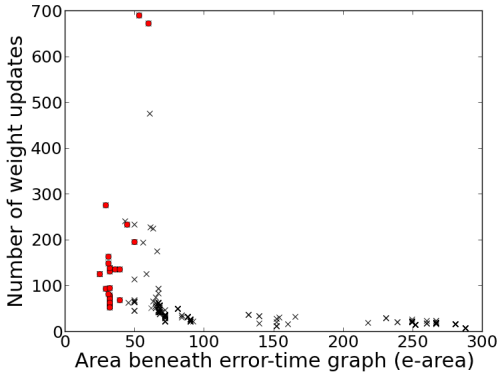
(b) NoNeg2



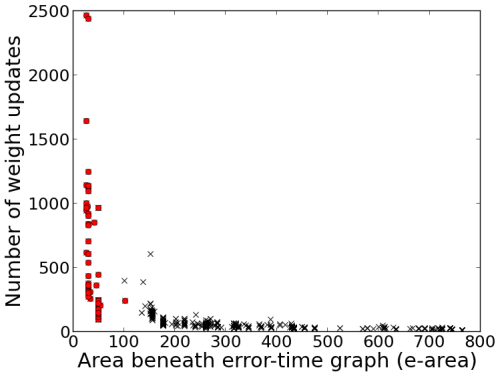
(c) NoNeg3



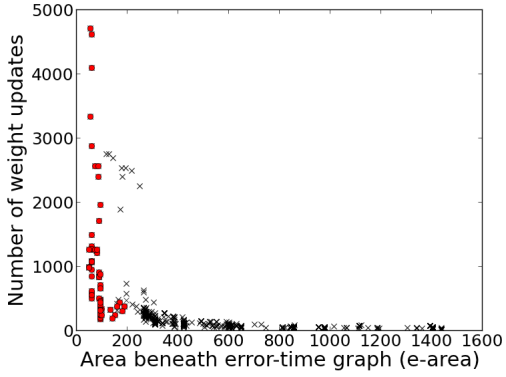
(d) NoNeg4



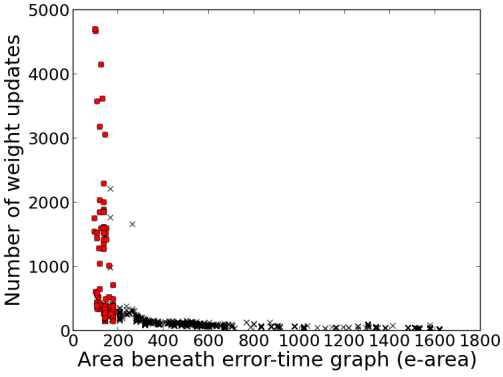
(e) Neg1



(f) Neg2



(g) Neg3



(h) Neg4

Figure 3.24. Final Pareto fronts for all 50 trials for each logic program. Red dots indicate zero-error networks that answer all questions correctly.

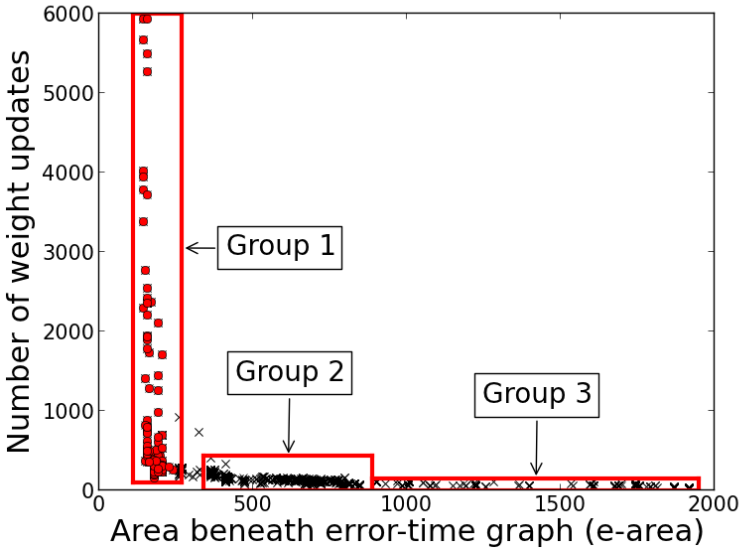


Figure 3.25. Three groups of genomes in the final Pareto fronts.

roughly separated into different partitions in objective space (fig. 3.25), although there is some overlap. Each of these groups had their own strategies for answering questions, which are discussed below.

Group 1 genomes (zero-error networks)

Members of this group can be identified by points marked by red dots in fig. 3.25. The networks developed by these genomes yield zero error, answering all training questions correctly. However, the genomes covered a range of values for the number of weight updates. All genomes sampled behave in the same way as those presented in section 3.2.3 in that they all limit the development of connections to nodes of the same function, and the more efficient genomes that yield a smaller number of updates do so by borrowing conditions from the learning algorithm and restricting development to active neurons, like genomes G3 and G4 in section 3.2.3.

Some genomes enforce the same conditions as G3 in that connections between SELF and P_INPUT are only produced when both nodes are active and are of the same function (fig. 3.26a). However, equivalent conditions or combinations of conditions also emerged that do not test for activity or function directly but produce the same behaviour. In genome 3.26b, condition 1 ensures that both nodes are firing in the same phase or not firing at all. Conditions 2 and 3 ensure that at least one of the nodes is firing because the value of $Time\_Window$ (the remaining time in the window of synchrony) for one must be greater than the other in order for the condition to be true. However because of condition 1, if one node is firing then both must be firing, therefore both SELF and P_INPUT are active. A phase value of -1 indicates no phase (the node is inactive), 0 indicates continuous, uninterrupted phase (the node is an enabler or collector) and a phase value greater than 0 indicates the firing time of a spike (the node is a role node). If condition 1 is true in the latter case, both nodes must be role nodes and are therefore nodes of the same function.

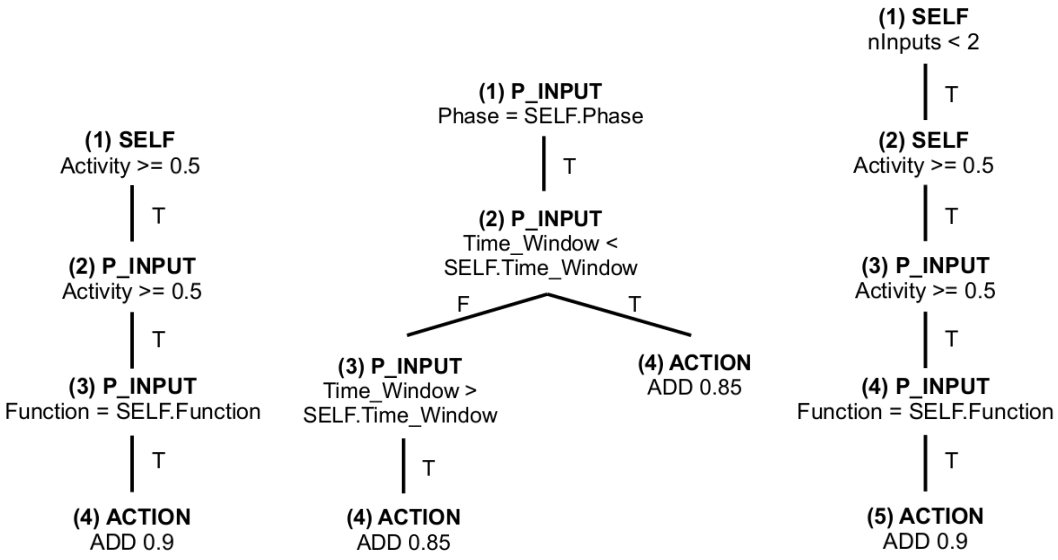


Figure 3.26. Genomes for constructing zero-error networks, sampled from the full set of 2695 from the final populations across all trials. All genomes in this figure behave like genome G3 from fig. 3.9c on page 82 in that they produce connections between active nodes of the same function. (a) does so using the same conditions, (b) does so with a different set of conditions, and (c) uses the same conditions but also limits the number of inputs to a node (condition 1).

In the case of a continuous phase condition 1 alone cannot determine whether the nodes are enablers or collectors. However if condition 2 is true, P_INPUT fires first, which is true when two collectors coactivate. If condition 3 is true, SELF fires first, which is true when two enablers coactivate. In summary, conditions 1 to 3 together, but not in isolation, ensure that SELF and P_INPUT are both active and of the same function.

The genome in fig. 3.26b restricts the number of connections even more than genome G3, since in order for a connection to be formed nodes must not only be coactive but must fire in the same phase (condition 1) and one before the other (conditions 2 and 3). These are also conditions for causal Hebbian learning. Genomes this restrictive are effectively performing the role of the learning algorithm before learning takes place. Given that minimising the effort of the learning algorithm was one of the objectives, the discovery of such genomes was inevitable. One of the goals of evolving SHRUTI genomes was to discover improvements to the model already proposed in section 3.2. Such improvements have been found in genomes like the one in fig. 3.26b, but only take the form of conditions that describe the learning algorithm.

Although restriction by activation and equal node function should enable a zero-error genome evolved for one logic program to develop networks for others, this may not always be the case if there are additional conditions defined in the genome that may be problematic for developing networks on previously unseen event sequences. For example, the genome in fig. 3.26c only allows each node to develop two input connections (condition 1). This may enable networks for smaller logic programs to develop so that they yield zero-error, but not larger programs. This issue is discussed further in section 3.4.3.

The fact that all zero-error genomes found in the search behave in the same way, albeit with

different representational mechanisms, suggests that the range of genotypic behaviours that enable the production of zero-error networks is very limited and must work in a very precise manner. Furthermore, the results support the claim made in section 3.2.4 that it is difficult to restrict the number of connections formed without enforcing conditions that mimic the learning algorithm. This supports the argument that in the SHRUTI model pre-existence of relational structures or observation of evidence for the relations they represent are the only means by which those structures can develop.

Although group 1 genomes yield zero-error networks for logics programs without conjunction, this is not the case in chapter 4 in which networks are developed for relations that do contain conjunction because the developed networks cannot correctly answer questions on conjunctive relations. Therefore it is important to note that the term *'group 1 genome'* refers to genomes that produce zero-error networks for programs without conjunction, and the term *'group 0 genome'* will be used in chapter 4 for genomes that develop networks that produce zero-error networks for programs that do contain conjunction.

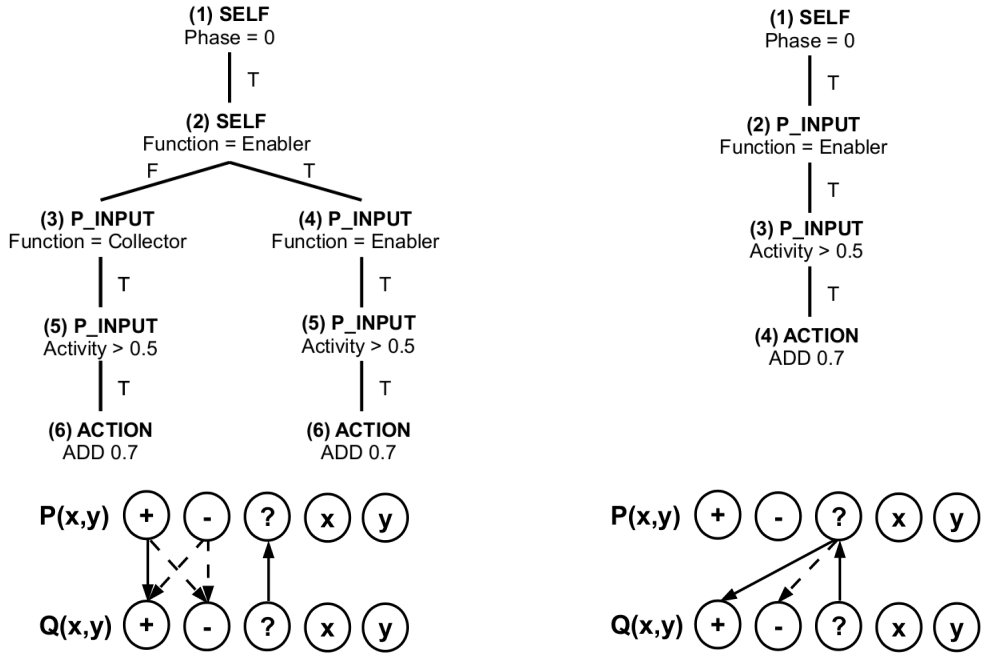
Before continuing, it is worth mentioning that the ability of the evolutionary search to discover genomes like these that restrict the number of connections and thus represent the minimal network structure required to represent a target logic program may have application in the field of knowledge extraction [51]. Knowledge extraction attempts to find meaning in trained neural networks. New neural networks could be evolved to behave in the same way as an existing neural network N from which knowledge is to be extracted, with the additional objective of minimising the number of connections in the new networks. The result would be a set of simplified equivalents to N that perform the same functionality but with a minimal or near-minimal number of connections. It would be easier to interpret the behaviour of the simplified networks and extract knowledge from them than it would be to do the same in N.

Group 2 genomes

Members of this group form dense clusters found next to the first group in the objective space for each logic program, as shown in fig. 3.25 on page 103. Networks in this group will always produce the same answer for any given predicate. That is, the answer given depends only on the queried predicate without reference to its arguments. For example, in table

Question	Expected answer	Given answer
$P(a, b)$	1,0	1,0
$Q(a, b)$	0,1	0,1
$R(a, b)$	1,0	0,0
$P(c, d)$	0,0	1,0
$Q(c, d)$	0,0	0,1
$R(c, d)$	0,0	0,0

Table 3.11. An example of the answering strategy for networks developed by group 2 genomes. The same answer is always given for any predicate. Here, questions relating to P are always answered [1,0] (true), questions on Q are always answered [0,1] (false), and questions on R are always answered [0,0] (unknown).



(a) Enabler-to-enabler and collector-to-collector connections are formed.

(b) Only enablers receive connections, either from collectors or other enablers.

Figure 3.27. Genomes for constructing group 2 networks, sampled from the final populations. In both cases no role-to-role connections are formed. Dotted lines in the relational network indicate connections weakened through the learning process. A phase of 0 refers to a continuous phase, as emitted by τ-and nodes such as enablers and collectors.

3.11, a network may always answer [1,0] (true) for predicate P, [0,1] (false) for predicate Q, or [0,0] (unknown) for predicate R. Any question for a predicate P with two arguments will always be interpreted as $P(x, y)$, which will always be true if the set of facts contains a positive instance of P, or will always be false if the set of facts contains a negative instance of P. To use a natural language example, consider the predicate $Own(x, y)$ and a knowledge base containing only one fact 'John owns the book' ($Own(John, Book)$). A group 2 network would assert any instance of Own as true, for example 'Mary owns the book' or 'Mary owns the ball', even though there is no information in the knowledge base from which these conclusions can be drawn.

At one extreme, all questions are answered 'true' or 'false' with no questions answered as 'unknown'. Such networks are produced by only developing connections between active enablers and collectors so that querying a predicate by activating its enabler leads to the automatic activation of one of its collectors. This is enforced by some set of conditions that ensures only enablers and collectors receive inputs (conditions 1 to 4 in fig. 3.27a and conditions 1 and 2 in fig. 3.27b). In all cases, connections between enablers are always formed, and one of the following is true:

Connections between collectors are formed (Fig. 3.27a): Collector and enabler connections are produced as they would be in a zero-error network, but there are no connections between role nodes to propagate role bindings. Given a relation $P(x, y) \rightarrow Q(x, y)$ and a fact $P(a, b)$, if $Q(c, d)$ is asked, the enabler of Q activates the enabler of P without propagating any role bindings. Therefore the question propagated to the

antecedent is $P(x, y)$, which always evaluates as true since $P(a, b)$ is true, even though this does not prove $Q(c, d)$.

Connections from enablers to collectors are formed (Fig. 3.27b): Consider again an arbitrary relation $P(x, y) \rightarrow Q(x, y)$. A collector of Q receives a connection from the enabler of P, which receives a connection from the enabler of Q. Therefore one of the collectors of Q is indirectly activated by its own enabler, and asking $Q(x, y)$ for any values of x and y will always answer true or false.

As e-area increases along this set, error tends to increase also, due to the correlation between the two statistics as explained in section 3.4.1. Observation of the output of these networks found that they also provided the same answer for any given predicate. However for greater error values, more predicates were always answered with 'unknown', up to and including networks that always answer 'true' or 'false' for only one of the predicates, the minimum required to mean that the network does not always answer 'unknown'. Anything less will result in all questions being answered 'unknown', the score being penalised to the maximum error and the network being classified as belonging to the third group. For all of these genomes, connections were again only formed between enablers and collectors, but there was some restriction on the number of inputs that SELF or P_INPUT could have before any new connections were created between them, therefore also restricting the total number of relations and questions that could be answered.

As with group 1 networks, equivalent sets of conditions to those in figure 3.27 will produce networks that behave in the same way. Also, connections are again limited by function in each case (even though role nodes are ignored) and occasionally also by conditions that mimic the learning algorithm, suggesting that few alternatives exist, if any at all.

Because connections are only produced between collectors and enablers, the number of connections and therefore the number of weight updates is kept relatively low. Such networks are therefore difficult to dominate with respect to the second objective, which explains why final Pareto fronts are heavily populated with members of this group.

Group 3 genomes (maximal-error networks)

These networks always answer [0,0] (unknown) and are therefore penalised with the maximum possible error. This happens owing to a complete lack of connections at the end of development, which happens for one of two reasons. The first reason is that either the genome contains no actions for creating new connections or any that do exist are never expressed. These are easily produced by randomly generating a nonsensical genome in the initial population or by mutating a group 1 or 2 genome. Group 1 and 2 genomes are extremely fragile in that only one mutation affecting a direct or indirect constraint on function equality, or an action of connection formation, could be enough to cause serious fault in the network.

The second cause is that any connections that are formed are removed later in the developmental process. Once again, some of these genomes may be mutations of group 1 or 2

genomes. In either case, the network yields some non-maximal error value for some period of time until all of its connections are removed and its error increases to maximum. This could be true of any error value before the connections are removed, and therefore the group contains a range of values for e-area.

With so few connections, the number of updates is also likely to be very low, even zero. Therefore the final Pareto fronts contain many group 3 networks for the same reason they contain many group 2 networks: they yield a small number of weight updates and so they are difficult to dominate with respect to the second objective.

Some other genomes which do not fit into these three groups do exist but they are relatively few in number. The lack of variety may possibly be attributed to two reasons. The first is that as argued before in section 3.2.4, the structure of SHRUTI networks is so specific that developing them by connecting nodes of equal function and possibly restricting development to active nodes may be the only means of constructing them. Group 1 genomes construct SHRUTI networks in precisely this way, and group 2 genomes are essentially group 1 genomes that do not form connections between role nodes. Any other genomes are either completely meaningless combinations of conditions and actions or destructive mutations of group 1 and 2 genomes. The other possible reason for the lack of variety is that even if alternative representations do exist, they cannot Pareto-dominate members of the three groups. The Pareto fronts are heavily populated by group 1 genomes by virtue of the fact that the networks they develop yield zero-error, and by group 2 and 3 genomes because the networks they develop include so few connections.

Small gaps between groups 1 and 2 were found in objective space for some logic programs (fig. 3.24). This is because of the difference between an error of zero and the minimum error that can be achieved by group 2 networks, and therefore also between the corresponding e-area values since the two statistics correlate. The gaps between groups 1 and 2 are more populated for larger networks because they yield a larger range of error values. The logic programs represented by larger networks contain a larger set of predicates. With more predicates, a greater number of questions can be asked and so a larger range of error values may be produced. Therefore the range of error and e-area values between the two groups is also greater.

The three groups described above form an ordering in that one group can be considered as an extension of another. Group 3 genomes contain no connections, but group 2 may be regarded as an extension of group 3 in that it produces connections between enablers and connections between collectors. Group 1 genomes also contain connections between enablers and collectors, but extend group 2 by also producing connections between role nodes. In this sense, the three genomes could be considered as three distinct evolutionary stages. As discussed in section 2.2, Piaget suggested a number of distinct stages of development [78], but modern psychologists would argue that the transition between those stages is more gradual and not so discrete [100]. Although the three groups of genome have been discovered through a nature-inspired evolutionary algorithm, these findings may have more biological plausibility if a greater number of transitional genomes could be found to

reflect a more gradual transition between the three distinct groups.

3.4.3. Performance on test data

Genomes from each trial were tested on event sequences and test questions from logic programs other than that logic program each genome was evolved to represent. Fig. 3.28 shows the number of trials from each logic program in the training data containing zero-error genomes that generalise to the event sequences and training questions from other logic programs on that all questions are answered correctly. Fig. 3.29 shows the proportion of zero-error genomes generated across all trials for which this is true.

Results from every training set found genomes that generalise to events and questions from other sets. However, genomes from the NoNeg data sets are less efficient at developing

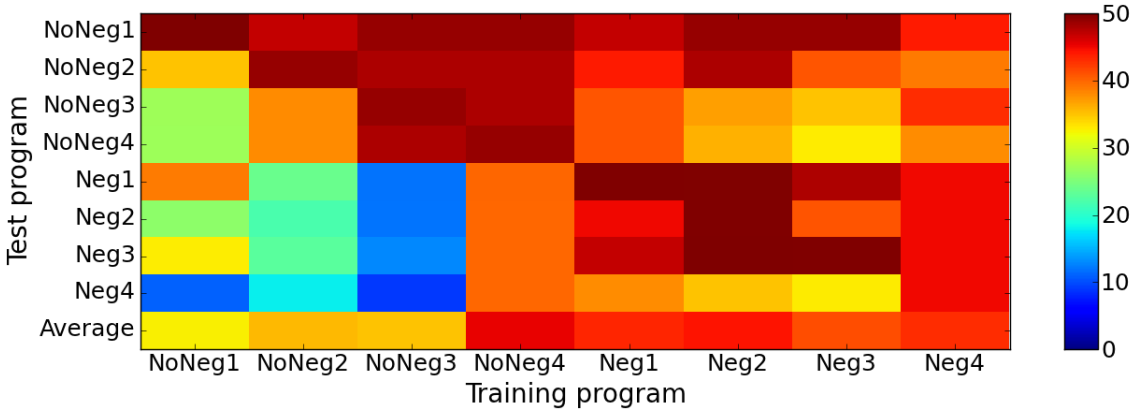


Figure 3.28. The number of trials for each logic program in the training data containing zero-error genomes that generalise to the event sequences and training questions from other logic programs. The colour of a square reflects the number of trials containing zero-error genomes evolved on the corresponding logic program on the x-axis that also develop zero-error networks for the corresponding logic program on the y-axis. For example, 39 trials (orange) evolved on NoNeg1 contain zero-error genomes that produce zero-error networks for Neg1.

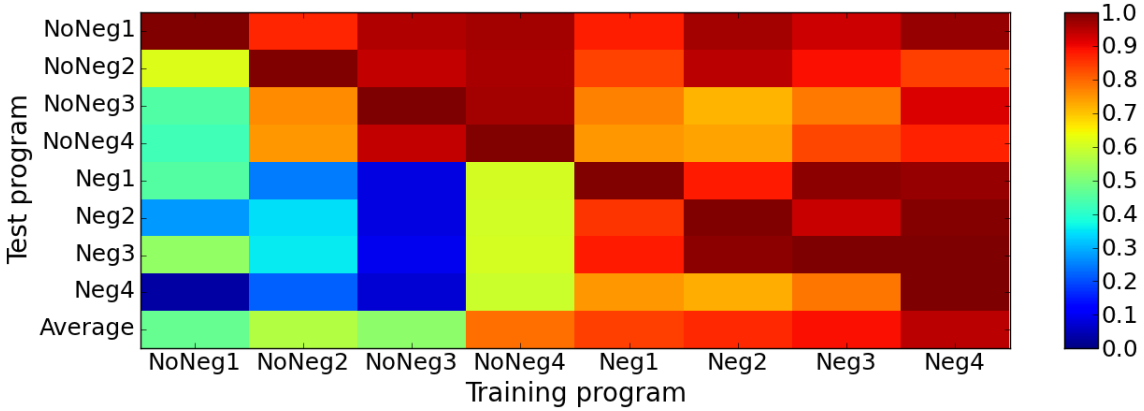
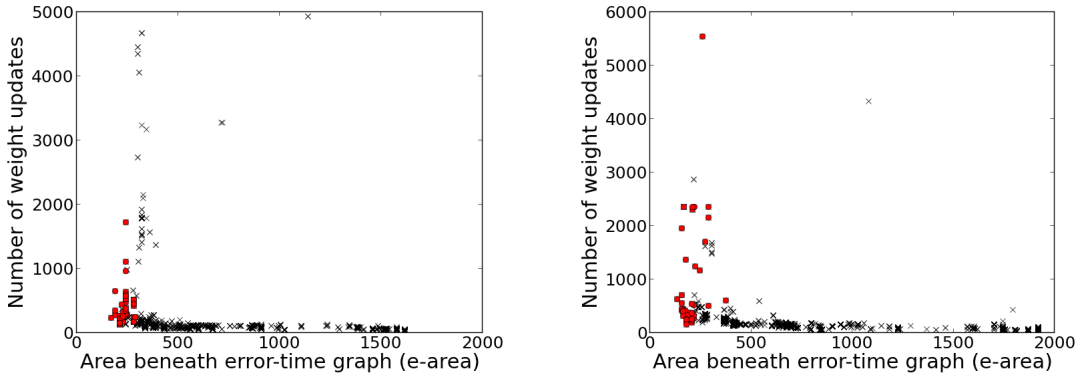


Figure 3.29. The proportion of zero-error genomes across all trials for each logic program that generalise to the event sequences and training questions from other logic programs. The colour of a square reflects the number of zero-error genomes evolved for the corresponding logic program on the x-axis that also develop zero-error genomes for the corresponding logic program on the y-axis. For example, very few (0.034, dark blue) of the zero-error genomes evolved on NoNeg1 develop zero-error networks for Neg4.



(a) NoNeg4 genomes producing networks for the Neg4 logic program

(b) Neg4 genomes producing networks for the NoNeg4 logic program

Figure 3.30. Pareto fronts produced when genomes evolved for one logic program are tested by developing networks for another.

networks for the Neg data sets than the Neg sets are at developing networks for the NoNeg data sets. This is because genomes evolved for the Neg sets are trained on relations containing positive and negative predicate instances and are therefore able to develop networks that are not brittle to the conflicting relations issue (section 3.1.2), whereas the NoNeg genomes are not exposed to such relations during training. Therefore the genomes evolved on Neg data generalise better.

Furthermore, as the size of the logic program used for training increases, the evolved genomes tend to be more generalisable as shown for the average results in fig. 3.29. Although the hypothesis was that a genome evolved to represent one logic program should generalise to others due to the fact that the same relational network structure is produced for all relations, this is not always the case if a genome evolved for a smaller logic program is presented with the training data for a larger logic program, which requires a larger network to represent it. Even though, as shown in section 3.4.2, all zero-error genomes connect nodes by function, some genomes contain additional conditions that restrict the development of conditions in such a way that they enable zero-error genomes to be developed for some event sequences but not others. For example, one of the genomes evolved to develop networks for NoNeg1 contains a condition that limits the number of inputs a node may have to two (fig. 3.26c, page 104), which probably resulted from minimising the number of weight updates and connections. This is not too restrictive for NoNeg1 because two inputs is the most that any collector node ever requires. However, the collector for $S(x, y)$ from NoNeg4 requires three inputs: one from each antecedent and one from the associated fact. Therefore, this genome evolved to represent NoNeg1 does not generalise to NoNeg4. In general, there is no guarantee that a genome evolved to represent a smaller logic program can develop a network that is able to represent a larger one, and therefore to evolve generalisable genomes it is better to evolve genomes on larger programs.

Figure 3.30 plots the objective values produced by genomes evolved for one program when generating networks and answering questions for test data. The plots closely resemble the original Pareto fronts in fig. 3.24 on page 102 and thereby demonstrate the generalisability of evolved genomes even further.

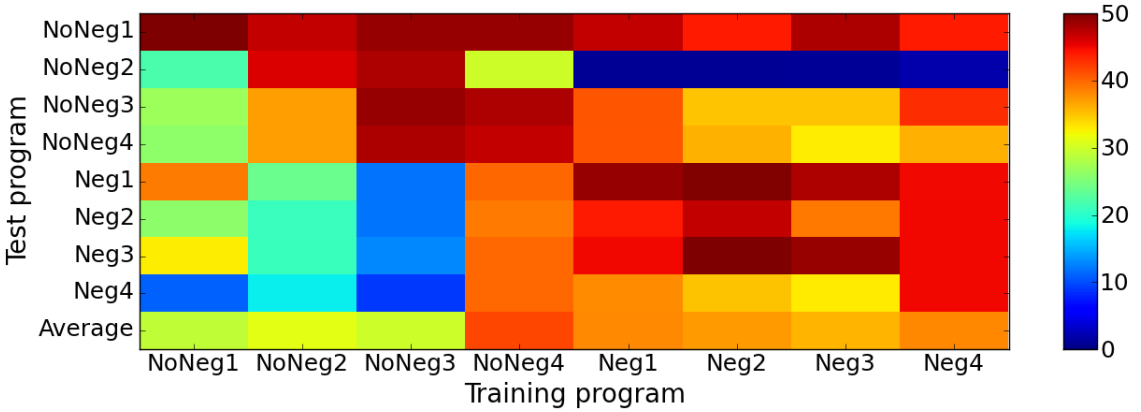


Figure 3.31. The number of trials for each logic program containing genomes that could generate zero-error networks for all possible questions in each test program.

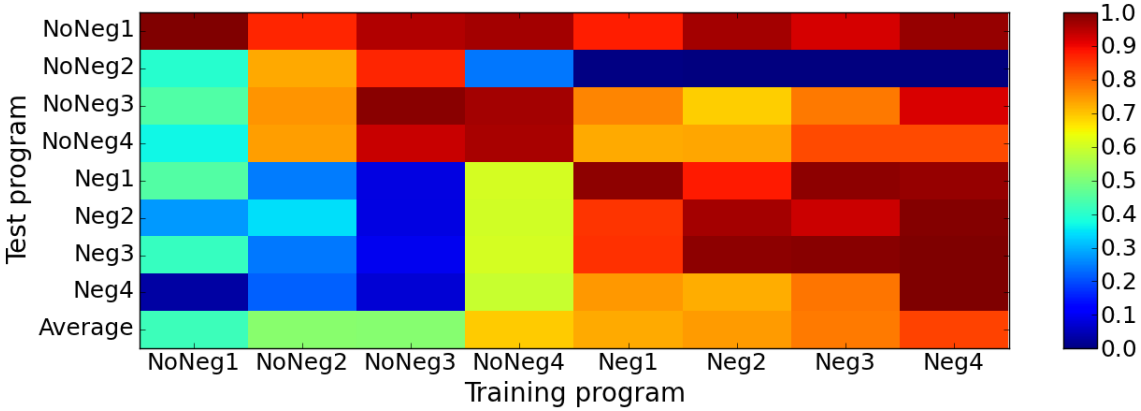
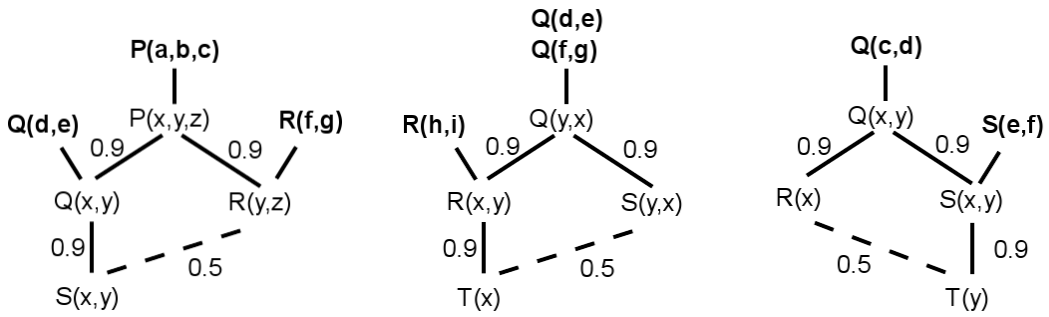


Figure 3.32. The proportion of zero-error genomes for each logic program that could generate zero-error networks for all possible questions in each test program.

The next experiment tested the generalisability of each genome to every possible question each logic program could be asked, including all possible unknowns. Fig. 3.31 shows the number of trials containing zero-error networks that generalised to all questions, and fig. 3.32 shows the proportion of zero-error networks for which this is true.

With the exception of NoNeg2, genomes evolved from most training sets develop networks that generalise very well to the full range of questions presentable to the logic program on which they were trained, and the proportion of genomes that generalise to all possible questions of other sets is almost as high as that of those that generalise to the training questions only (fig. 3.32). This supports the argument that as long as every relation is represented by at least one question in the training set, only small sets of training questions are necessary to assess the fitness of a network. If a trained network can answer one question from a relation correctly, most of the time, it will be able to answer them all.

However, few genomes generalised well to the NoNeg2 data set. In particular, the networks they developed would incorrectly answer the question $S(x, g)$ as [1,0] (true), when it should be answered as [0,0] (unknown). This occurs because some genomes develop networks that learn the relation $R(y, z) \rightarrow S(x, y)$, which is not actually a relation in the NoNeg2 logic program. The problem occurs when this undesired relation is learned with weights of 0.5 or slightly greater (Fig. 3.33), just enough to cross the activation threshold of the



(a) NoNeg2 (b) NoNeg3 (c) NoNeg4

Figure 3.33. Target relational structures of the form $A \rightarrow B, A \rightarrow C, C \rightarrow D$ when developed using Neg genomes, a bias towards strengthening relations ($hInc = 1.25$) and fixed event sequences. Undesired relations of the form $B \rightarrow D$ are incorrectly learned such that the weight values of connections that represent those relations are just strong enough to cross a node's threshold. Facts associated with relations are printed in bold.

nodes in the relation and propagate the truth of $R(f, g)$ to $S(x, y)$. This relation gains strength whenever evidence for $R(y, z)$ is observed at the same time as $Q(x, y)$. $S(x, y)$ is a consequence of $Q(x, y)$ and therefore whenever $Q(x, y)$ and $R(y, z)$ are observed together, $S(x, y)$ is also observed to follow $R(y, z)$ as well as $Q(x, y)$, suggesting that $S(x, y)$ is a consequence of $R(y, z)$ even though it is not. Evidence of $R(y, z)$ occurring without $S(x, y)$ is presented to weaken the relation, but only enough to settle the relation's weights at approximately 0.4 in the NoNeg trials, just below the threshold and therefore sufficient for the relation between the two predicates to be interpreted as non-existent. This works because NoNeg genomes run with no bias towards strengthening relations ($hInc = 1$, see equation 3.1 and its explanation on page 76). However, because the Neg genomes are run with the bias towards strengthening relations ($hInc = 1.25$) in order to overcome the conflicting relations problem, the weights are more likely to settle at higher values, just above 0.5, after training. Therefore, the Neg genomes are highly likely to develop networks that incorrectly learn the relation $R(y, z) \rightarrow S(x, y)$. Similar relations are learned by Neg networks on $NoNeg3$ (fig. 3.33b) and $NoNeg4$ (fig. 3.33c) training sequences also. However no incorrect conclusions are made because the antecedents in these unwanted relations, $S(y, x)$ and $R(x)$ respectively, have no facts associated with them. Were these experiments to be repeated, the fixed training sequences would be constructed such that better, less ambiguous evidence for relations such as $R(y, z) \rightarrow S(x, y)$ in NoNeg2 would be presented. This is in fact the case for the probabilistic event sequences, which do enable NoNeg2 to be learned properly. This is discussed in more detail below.

Nonetheless, the results show that most genomes that evolved to develop networks that answer all their training questions correctly generalise well to unseen event sequences and test questions.

The next experiment tested generalisability to probabilistic event sequences generated according to the probabilities listed in appendix A. Tables 3.34 and 3.35 present the results of testing up to five genomes, chosen at random from the Pareto front of each trial, on 10 random seeds of 500 observations and all possible test questions. As argued in section 3.1.2, the learning model does not support negation in a probabilistic event sequence. Therefore,

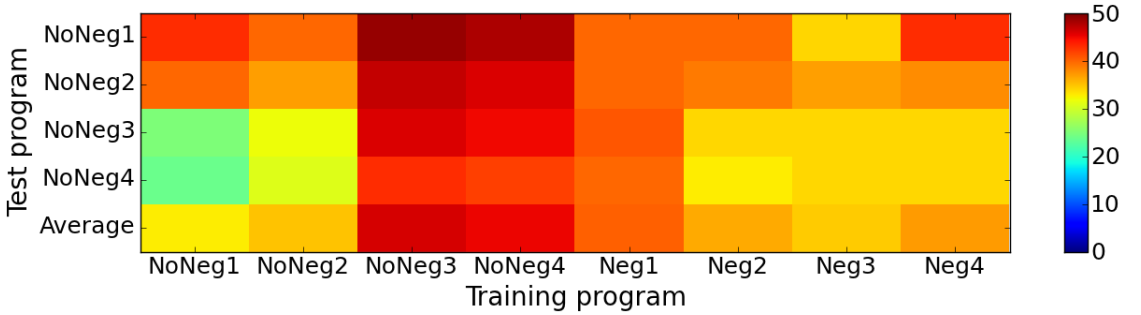


Figure 3.34. The number of trials for each training program containing zero-error genomes that generalise to probabilistic event sequences for each test program, based on random samples of up to five zero-error genomes for each trial.

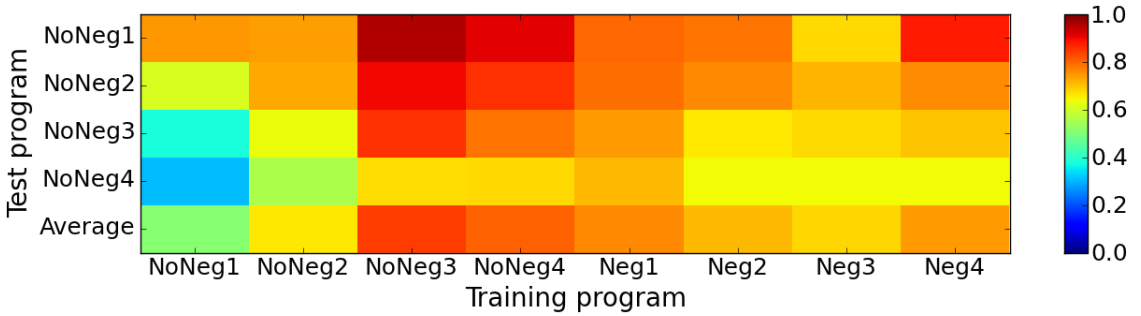


Figure 3.35. The proportion of zero-error genomes sampled over all trials for each training program that generalise to probabilistic event sequences for each test program, based on random samples of up to five zero-error genomes for each trial.

as expected, none of the evolved genomes developed zero-error networks for the Neg data sets using the probabilistic sequences and so results for these are excluded for brevity. However genomes from all trials were able to generate some zero-error networks for the NoNeg probabilistic sequences, although not as successfully as tests on fixed sequences. Although the proportion of successful generalisations is below 70% for genomes trained on NoNeg1, NoNeg2 and Neg3, that of genomes trained on other sets is reasonably high considering that they were not trained on probabilistic event sequences (for reasons given on page 77).

Once again, where genomes do not generalise successfully, this is due to additional conditions in the genome that may not impose problems on fixed event sequences but do on probabilistic ones. Consider the example of the NoNeg1 genome that restricts the number of input connections for a node to two (fig. 3.26c, page 104). If evidence for $P(x, y) \rightarrow Q(x, y)$ is observed before that of $Q(x, y) \rightarrow R(x, y)$, as in the fixed event sequence, $+Q$ will receive input from $+P$, as is necessary to represent the target relation. However, if, as happens in roughly half the probabilistic event sequences, evidence for $Q(x, y) \rightarrow R(x, y)$ is observed before that of $P(x, y) \rightarrow Q(x, y)$, $+Q$ will instead receive an input connection from $+R$ first. Along with the input from the fact $Q(e, f)$, $+Q$ now has two input connections and no more space for the required input connection from $+P$ to be formed.

It is worth noting that results for generalisation to NoNeg2 are actually better when using the probabilistic event sequences than when using the fixed sequences, because

Relation	Probability	Connection	Average weight				
			1	2	3	4	5
$P(x,y) \rightarrow Q(x,y)$	0.9	+P → +Q	0.922	0.886	0.915	0.890	0.871
		?Q → ?P	0.902	0.872	0.930	0.895	0.862
$Q(x,y) \rightarrow R(x)$	0.8	+Q → +R	0.813	0.829	0.842	0.875	0.811
		?R → ?Q	0.799	0.805	0.829	0.863	0.801
$Q(x,y) \rightarrow S(x,y)$	0.75	+Q → +S	0.800	0.782	0.766	0.801	0.779
		?S → ?Q	0.793	0.791	0.776	0.794	0.783
$S(x,y) \rightarrow T(y)$	0.85	+S → +T	0.890	0.858	0.882	0.884	0.875
		?T → ?S	0.895	0.853	0.887	0.893	0.864
$U(x,y,z) \rightarrow V(x,y,z)$	0.7	+U → +V	0.781	0.746	0.775	0.753	0.683
		?V → ?U	0.754	0.738	0.780	0.721	0.695
$V(x,y,z) \rightarrow S(x,y)$	0.8	+V → +S	0.858	0.837	0.800	0.816	0.819
		?S → ?V	0.834	0.821	0.803	0.824	0.803

Table 3.12. Final weights of connections representing relations in NoNeg4, following genetic development and training on probabilistic event sequences. The learned weight values are averaged over 10 trials for 5 zero-error genomes produced through evolution, and are similar to the probabilities of the corresponding relations.

the probabilistic event sequence provides less support for $R(y,z) \rightarrow S(x,y)$ in that the predicate $R(y,z)$ occurs more frequently without $S(x,y)$ than it does in the fixed event sequence. Therefore the problem of genomes not generalising well to NoNeg2 in previous experiments was not owed to the evolved genomes but to the evidence presented by the fixed event sequence.

Finally, a few of the evolved zero-error genomes were sampled to observe how well connection weights learned for developed networks reflected the probabilities used to generate event sequences. Table 3.12 shows learned connection weights for 5 zero-error genomes, developing networks for NoNeg4. As has already been argued, logic programs from the Neg datasets cannot be learned from probabilistic event sequences because the weights of enabler-to-enabler and role-to-role connections do not reflect the probabilities of the corresponding relations, owing to the conflicting relations problem. However, the results in table 3.12 demonstrate that for NoNeg logic programs, evolved zero-error genomes develop networks in which learned connection weights do reflect the probabilities of the corresponding relations.

3.5. Larger Predicates

All experiments performed thus far were performed on logic programs with predicates containing no more than three arguments, in order to limit execution time when evaluating developed networks during the evolutionary search (see page 71 for a defence of this decision). This enables no more than three argument bindings at any one time, even though the human brain is believed to be capable of performing 7 ± 2 bindings at any one time [95]. Given that the evolved genomes represent connections between roles nodes (which represent arguments) of two predicate clusters regardless of how many arguments those predicates have, it is argued that the genomes are just as capable of developing networks for relations containing seven arguments per predicate, or any number of arguments for

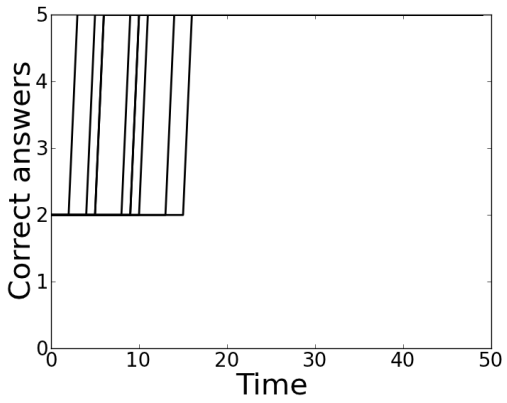


Figure 3.36. The number of correct answers over time as genome G1 develops a network for the single relation $P(t, u, v, w, x, y, z) \rightarrow Q(t, u, v, w, x, y, z)$, with a probability of 0.9. 10 trials are performed, with a new event sequence generated in each case. All questions are answered correctly.

Logic program	NoNeg1	NoNeg2	NoNeg3	NoNeg4	Neg1	Neg2	Neg3	Neg4
Generalisations to 7-argument relation	0.910	0.825	0.860	0.971	0.969	0.920	0.844	0.928

Table 3.13. The normalised number of zero-error genomes evolved on each logic program that generate a zero-error network for the 7-argument relation $P(t, u, v, w, x, y, z) \rightarrow Q(t, u, v, w, x, y, z)$. The same fixed event sequence, containing multiple repetitions of the sequence $(P(t, u, v, w, x, y, z), Q(t, u, v, w, x, y, z))$, was used in each case.

that matter, as they are for relations containing up to three arguments per predicate.

Genome G1 (fig 3.9a, page 82) and all evolved genomes were tested on the development of a relational structure for a single relation that contains seven arguments in both the antecedent and the consequent. The relation used was $P(t, u, v, w, x, y, z) \rightarrow Q(t, u, v, w, x, y, z)$, with probability 0.9, facts $P(a, b, c, d, e, f, g)$ and $P(h, i, j, k, l, m, n)$, questions $Q(a, b, c, d, e, f, g)$ and $Q(h, i, j, k, l, m, n)$ expected to be answered 'true', and questions $Q(c, i, f, h, g, b, i)$, $Q(i, h, c, n, m, j, a)$ and $Q(g, a, b, b, h, i, d)$ expected to be answered 'unknown'.

Fig. 3.36 shows the number of questions answered correctly over time as G1 develops a network for the seven-argument relation. 10 trials were performed, with a new event sequence generated in each case, according to the probability of the relation being true (0.9). All five questions are answered correctly before twenty observations have been made, thus demonstrating that G1 generalises to relations in which predicates contain up to seven arguments.

Table 3.13 shows the proportion of zero-error genomes evolved for each logic program that generate zero-error networks for the relation $P(t, u, v, w, x, y, z) \rightarrow Q(t, u, v, w, x, y, z)$. For example, 91% of the zero-error genomes evolved for NoNeg1 produce networks for this relation that are able to answer all five questions correctly. In most cases, the genomes generalise reasonably well, but performance may have been improved if the original training data for which genomes were evolved contained logic programs in which predicates had seven arguments or more.

These results demonstrate that SHRUTI genomes can scale to a biologically plausible number of variable bindings, which is believed to be 7 ± 2 [95]. This has been demonstrated for only one relation, and evolution of larger logic programs which contain relations of this size was not attempted due to the computational expense of doing so (see page 71). This remains a matter for future work. However it has already been argued throughout this chapter, particularly in section 3.2.3, that SHRUTI genomes scale to logic programs of any size because the genome represents a network structure for an arbitrary logical relation and produces it for each relation in a logic program. Given that the genomes can generate a relational structure for up to seven predicate arguments, they should be equally as capable of developing larger networks containing multiple instances of this structure.

3.6. Discussion

This discussion section summarises the findings of this chapter, of which there are two main conclusions. The first is that as SHRUTI's developers suggest [96, 120], owing to the use of repeated substructures in SHRUTI networks, these networks can be pre-organised in a scalable, biologically plausible way by using artificial genomes that employ indirect encoding to direct the growth of connections between nodes in predicate structures. The genomes do not produce the nodes themselves, but this is addressed later in chapter 4. The second main conclusion of this chapter is that these SHRUTI genomes can be produced in a biologically plausible way by an evolutionary search. Both of these conclusions support the biological plausibility of the SHRUTI model as a whole. However, according to the definition of biological plausibility presented in section 2.1.7, the biological plausibility of this developmental SHRUTI model is limited in that it is still an abstraction that captures only some aspects of real neurogenesis, which are summarised below on page 118.

The genomes designed manually and most genomes discovered through evolution generalise very well to unseen event sequences and test questions. Test sets vary in both the size of the logic programs themselves and the questions that may be asked of them, demonstrating that the genomes are scalable in the sense that the size of the genotype is independent of that of the phenotype. Genomes that support development according to probabilistic event sequences were also found. It was hypothesised that genomes evolved on fixed event sequences would generalise better to probabilistic event sequences than they did. However, because they slightly underperformed compared to what was expected, genomes should be evolved on probabilistic event sequences in future work. Nonetheless, most evolved genomes still generalised well to noisy, probabilistic event sequences. Although generalisation to probabilistic event sequences for logic programs containing negation was completely impossible, this is owed to a more fundamental problem with the causal Hebbian learning algorithm and not the genome model itself.

The generalisability of SHRUTI genomes is owed to the very nature of indirect encoding: the representation of repeatable substructures that are scalable to structures of different sizes (section 2.5). In particular, the substructure in question here is a network representation of a logical relation that is transferable to any logical relation in any logic program.

It is a relational substructure, and not any complete network, that is evolved. In other words, the evolved genomes represent the general relational structure (section 2.3.3) as it is applied to SHRUTI networks that do not involve conjunction. The bare minimum required to discover such generalisable encodings in an evolutionary process is a question set that contains at least one question that is representative of each logical relation.

One of the aims was to discover alternatives or improvements to the genome model in section 3.2. Although conditions in evolved zero-error genomes were sometimes different from those in section 3.2, the resulting behaviour that they enforced was equivalent. They all developed zero-error networks by restricting the development of connections to nodes of the same function, and in the case of genomes that improved efficiency, restricted connections further by copying elements of the learning algorithm. The only improvements discovered were also rules similar to those in the learning algorithm, for example genomes which restrict connections to active nodes firing in the same phase, just as causal Hebbian learning requires that role nodes fire in the same phase for a connection weight to strengthen. Work in the background section also found that the discovery of optimal weight values during evolution resulted in fitter genomes, again due to taking some work from the learning algorithm (section 2.5.1).

No alternatives were found, suggesting that the means of representing such SHRUTI networks in a genotype is limited and supporting the argument put forward in section 3.2.4 that because of the very specific nature of SHRUTI's localist structure, neural representations of target relations must either pre-exist or be created upon observation, thus copying aspects of the learning algorithm. The genomes only had a limited set of conditions with which to work and further conditions may allow the search algorithm to discover different networks. However all attributes that a node or connection can possess as far as SHRUTI relations are concerned are included in the model. Even if new conditions are introduced, the fact remains that the structure for a relation must exist in order for that relation to be learned in the first place because it is the structures themselves that form the representations. All the learning algorithm does is determine which of these relations are true and which are false.

Although development produces repeatable substructures in biological systems, it is not clear whether or not each of those substructures forms a specific representation as they do in SHRUTI. Given that neural representation is more widely believed to be distributed and not localist [80], such fully localist representations as used by SHRUTI are unlikely. A distributed approach would be one in which within any substructure, or even between substructures, any individual connection could participate in the representation of multiple observed relations. Furthermore, unlike in the SHRUTI genomes, biological development does not use learning to influence early (pre-natal) stages of development and it is likely that activity-dependent development supports learning without making it redundant. More biologically plausible behaviour could possibly be realised in a distributed representation of SHRUTI. Similar to the information processing theory of cognitive development [69, 100], development in a distributed system could support learning by providing new connections when the system becomes saturated with knowledge or error becomes

high, improving storage capacity and connectivity rather than building a representation of what needs to be learned. Furthermore, the information processing theory of development supports the more biologically plausible idea of a more continuous developmental process than is suggested by the discrete nature of the three groups of genomes presented in section 3.4.2. For these reasons, possible distributed representations of SHRUTI are worthy of further investigation. A discussion on this idea as a research direction, complete with some preliminary results, is provided later in chapter 5.

The genome itself exhibits some biological plausibility. In particular, indirect encoding is closer than direct encoding to the means by which biological genomes produce organisms, that is, by providing a set of instructions for the gradual development of the phenotype [22, 103, 116]. This is implemented by the biological process of gene regulation [116]. Also, gene expression in the SHRUTI genome also involves introns and exons, just as biological gene expression does. However, gene expression in SHRUTI is still an abstraction, and real biological networks are much more complex. Furthermore, the model of gene regulation used in the SHRUTI genome is a tree structure, and less like a network as in biology. Another biologically plausible trait of the system is the involvement of activity-dependent development [33], although once again due to the abstract nature of the genome the biological plausibility of this could be improved. While activity-dependent development is known to trigger the development of connections in neural development, it is unlikely to be as immediate and direct as it is in the SHRUTI genome. The evolutionary algorithm applied to the evolution of the SHRUTI genomes, including the crossover and mutation operators tailored for the particular genome representation, are also abstractions of biological processes. In summary, the SHRUTI genomes and the means through which they are evolved are biologically plausible in the sense defined in section 2.1.7 in that the model is a high-level abstraction of natural development that is not claimed to be a perfect copy, but captures some aspects of natural development in an attempt to understand how neural representations of reasoning could evolve and develop. The use of such abstraction in the genetic model was chosen because SHRUTI itself is also an abstraction, and therefore less abstract, lower-level genetic models could be explored as the biological plausibility of SHRUTI is improved.

Regardless of the extent to which biological plausibility can be claimed, the biological plausibility of the SHRUTI model has nonetheless been improved by introducing a developmental model as suggested by the SHRUTI developers, and by showing that these genomes can be produced through evolution. However, biological plausibility has only been improved with respect to connections between neurons and the representation of relations that do not involve conjunction. To support biological plausibility further, the next step was to generalise the ideas from this chapter to other structures in SHRUTI.

4. Evolution with recruitment learning

The previous chapter introduced a genome for constructing connections between nodes in SHRUTI predicate clusters that enabled Hebbian learning of relations between those predicates, and demonstrated that similar genomes could be produced through an evolutionary process. This supports the biological plausibility of these relational structures in SHRUTI by demonstrating that they can be represented in a biologically plausible way, as argued by SHRUTI's developers [96, 120], and that they can be discovered through the biologically plausible process of evolution. These relations were *non-conjunctive* in that they did not contain conjunction of antecedents, for example $A \wedge B \rightarrow C$ (a *conjunctive* relation). Non-conjunctive relations are but a small component of the overall SHRUTI architecture and in order to further support the biological plausibility of SHRUTI and the arguments of its developers that its repeated substructures can be represented by indirect encodings, the representation and evolution of further SHRUTI structures must be explored. Therefore this chapter extends the genome representation to enable the construction of mediator structures and fact structures, both of which were introduced in section 2.3.5. Mediator structures are intermediate clusters of nodes between antecedent and consequent predicate clusters that enable conjunctive relations to be represented by SHRUTI and also enable type restrictions on relations in general. Fact structures represent episodic facts as static bindings between entities and predicate arguments. Both mediator and fact structures are learned through recruitment learning (page 51, [93]). In order to enable the development and evolution of these structures, the updated genome model presented in this chapter now permits the development and evolution of nodes in a network, and not only the connections between them, as in chapter 3.

The findings of this chapter are summarised as follows. Representation of both mediator and fact structures using indirect encoding was successful, thus supporting the claims of SHRUTI's developers with respect to biological plausibility as far as genotypic representation is concerned. However, evolutionary searches were unable to rediscover these mediator and fact genomes as they were able to discover genomes for the non-conjunctive relations in the previous chapter, suggesting that SHRUTI networks in their current form are only evolvable to a certain extent. A variety of experiments were performed to explore the fitness landscape and show that it is not amenable to an evolutionary search. Based on this understanding of the fitness landscape, future work can explore more evolvable alternatives to the SHRUTI model.

More specifically, results show that these more complex SHRUTI structures require a number of necessary components to be discovered together and not individually before they can have any influence on objective fitness. Furthermore, the fitness landscape is

heavily populated by genomes that produce empty networks that always answer 'unknown' to any questions they are presented with, like the group 3 networks from the previous chapter. As a consequence, the fitness landscape is very difficult to explore and contains insufficient information to help an evolutionary search discover changes in genotype space that yield improvements in objective space. The genomes provide very explicit descriptions of SHRUTI's structures and therefore the requirement for multiple genetic components to be in place together in order for fitness to be improved is owed to the fact that same is true of SHRUTI networks themselves; multiple network components must all be in place together before they can positively affect the output of the network. This is most likely because SHRUTI uses fully localist representation that assigns very specific, necessary roles to different nodes. The behaviour of SHRUTI networks is very discrete and it may be that a more distributed, continuous alternative is required in order to produce an evolvable model of neural-symbolic reasoning. This may allow individual discoveries to make their own contributions to objective fitness so that changes in objective fitness reflect useful changes in genotypic space more accurately.

The implementation of recruitment learning used in the experiments described in this chapter is explained in section 4.1. Section 4.2 describes changes to the genome model that enable the construction of mediator and fact structures through the development of new nodes. The mediator and fact structures are discussed in sections 4.3 and 4.4 respectively. Both of these sections describe necessary changes to the learning algorithm, the genotypic representation of the corresponding structures, and the results of attempting to rediscover the genomes through evolution. The discussion in section 4.5 summarises the findings and makes a case for exploring more distributed models in the search for evolvable neural-symbolic networks.

4.1. Recruitment learning

A basic model of recruitment learning (page 51) is implemented for the experiments presented in this chapter. A recruited node is one that participates in the propagation of activation through the network. At the beginning of development only entity nodes and nodes in predicate clusters are recruited, and a node becomes recruited when any connections it shares with existing recruited nodes have gained sufficient strength through a

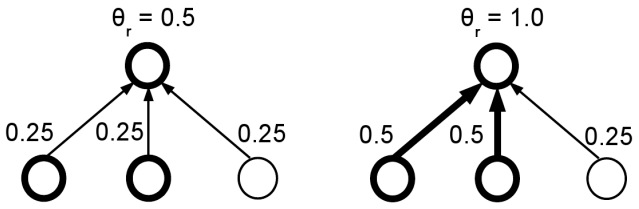


Figure 4.1. Recruitment with multiple inputs. Connections from the first two input nodes gain strength because their combined weighted activation crosses the recruitment threshold (θ_r) of the output node when activated, and therefore participate in its activation. Each time the recruitment threshold is crossed, it is increased to a value equal to the sum of the contributing input weights. Bold circles represent active nodes and bold lines represent connections gaining strength.

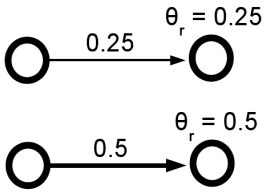


Figure 4.2. Recruitment with only one input. Only one input activation is required to activate the output node.

hInc	Numerator when increasing weight through causal Hebbian learning.
hDec	Numerator when decreasing weight through causal Hebbian learning.
rInc	Numerator when increasing weight through recruitment learning.

Table 4.1. Learning parameters in the updated learning model. Each parameter is used as the numerator when calculating learning rate α in equation 3.1.

model of Long-Term Potentiation [93]. At the beginning of development, Each node has a recruitment threshold θ_r, and when this is crossed connection weights of any inputs which participated in providing the necessary activation are strengthened (fig. 4.1). The recruitment threshold is initially low in order to allow initially weak sets of input connections to cross it, but is increased to a value equal to the sum of contributing input weights each time it is crossed. In the mediator and fact structures, as can be seen in sections 4.3 and 4.4 respectively, some of the nodes to be recruited contain only one input due to the highly localist nature of SHRUTI. These nodes become recruited as soon as their only inputs are activated (fig. 4.2).

Although weights are strengthened through Long-Term Potentiation (LTP), this model does not include Long-Term Depression (LTD), in which an input connection to an active post-synaptic node weakens when that pre-synaptic node is inactive [93]. For all recruitable nodes in the mediator and fact structures, recruitment thresholds are set so that the post-synaptic node only activates when all of its pre-synaptic nodes do. A post-synaptic node will never be active without the activation of all of its pre-synaptic nodes, and therefore the conditions for LTD never occur. Furthermore, the inclusion of LTD was not important for the evolvability of the mediator or fact structures because it is not the learning mechanism that is evolved, but the structures in which learning takes place.

In addition to a pair of learning parameters used in causal Hebbian learning (*hInc* and *hDec*, page 76), nodes now have an extra parameter *rInc* for recruitment learning. Table 4.1 summarises the full set of learning parameters. Whereas *hInc*, which supports the learning of relations, has a counterpart in the form of *hDec* for unlearning relations, *rInc* has no counterpart *rDec* since Long-Term Depression is excluded from this model.

4.2. Genome model

This section describes a number of changes to the genome model that were required in order to produce genomes for mediator and fact structures in sections 4.3 and 4.4. An understanding of the original genome model and how it is used to produce SHRUTI net-

Node function	Type	Chem1	Chem2
Collector (+ / -)	τ-and	0	1
Enabler (?)	τ-and	1	0
Role node n	m-ρ	0	f(n)*
Entity node	m-ρ	1	0

*f(n) = 0.81 if n =1, 0.9 × role_node(n-1).Chem2 otherwise

Table 4.2. Chemical levels and node types assigned to different nodes present before development. n refers to the position of an argument within the corresponding predicate. The choice of values for f(n) is explained in the text.

works is necessary to understand these changes (section 3.2.2). Particular adaptations that add to the biological plausibility of the model include the diffusion of chemical neuromodulators to enable cell differentiation [116] and the development of SHRUTI neurons themselves in addition to the connections between them [1, 12, 123].

Firstly, entity nodes, fact nodes, mediator nodes and inhibitory nodes could now all participate in the developmental process, and conditions testing node function had to account for these in addition to enablers, collectors and role nodes. As explained in section 2.3.5 (table 2.3), mediator and inhibitory nodes break down into further sub-categories which are regarded as individual functions by the genome. Mediator node functions can be classified as either 'Truth' mediators (*medTr*), which propagate truth by propagating activation between the collectors of predicate clusters, or 'question' mediators (*medQu*), which propagate queries by passing activation between enablers or between role nodes. medQu nodes may therefore be of the τ-and or m-ρ node types (section 2.3.5, table 2.2) depending on whether they mediate between enablers (τ-and) or role nodes (m-ρ). An example of a mediator structure can be found in fig. 2.9 on page 44. Each fact circuit (fig. 2.8, page 44) contains two arrays of inhibitor nodes that perform one of two functions and therefore they are classified as such. The first array of inhibitory nodes (*inhib1*) test argument-entity bindings, and the second (*inhib2*) blocks the activation of the fact node if the bindings at inhib1 are incorrect. In summary, the model now accounts for all nine node functions listed in table 2.3.

SHRUTI nodes in this new model now contain chemical neuromodulators which may be passed between nodes during development and help to determine a node's position relative to other nodes, as seen in some of the literature on biochemical methods of artificial development discussed in the section 2.5.2 [23, 24, 31, 32, 47, 57]. Before development, only enabler, collector, role and entity nodes pre-exist because they form predicate clusters and entities, from which other structures (facts and mediators) are produced through development. Each of the initial nodes must be prescribed with a set of chemical levels. Chemical levels are assigned such that nodes of each function contain a unique combination of node type (τ-and or m-ρ) and chemical level, as shown in table 4.2. However such information does not necessarily serve as a unique identifier for each node function because nodes of different functions but identical node types and chemical levels may emerge during development. Within a set of role nodes for each individual predicate, the initial amount of chemical 2 assigned depends on the position of the argument represented by that node. For example, for $P(x, y, z)$, chemical levels of 0.81, 0.729 and 0.6561 are assigned to x, y

Neuron conditions	
Activity	Activity level.
Phase	Current phase.
Time_Window	Remaining time before window of synchrony expires.
Function	Node function: ena, col, role, ent, fact, inhib1, inhib2, medTr or medQu.
Chemical 1/2	Amount of each chemical.
nInputs **nOutputs**	Total number of inputs or outputs.

Connection conditions	
Weight_I **Weight_O**	Weight of connection from E_CON to SELF (Weight_I) or from SELF to E_CON (Weight_O).
nUpdates_I **nUpdates_O**	Number of times connection weight from E_CON to SELF (nUpdates_I) or from SELF to E_CON (nUpdates_O) has been updated.
nIndirect_I **nIndirect_O**	Number of indirect connections (paths separated by one node) from P_CON/E_CON to SELF (nIndirect_I) or from SELF to P_CON/E_CON (nIndirect_O).

Table 4.3. Conditions in the genome

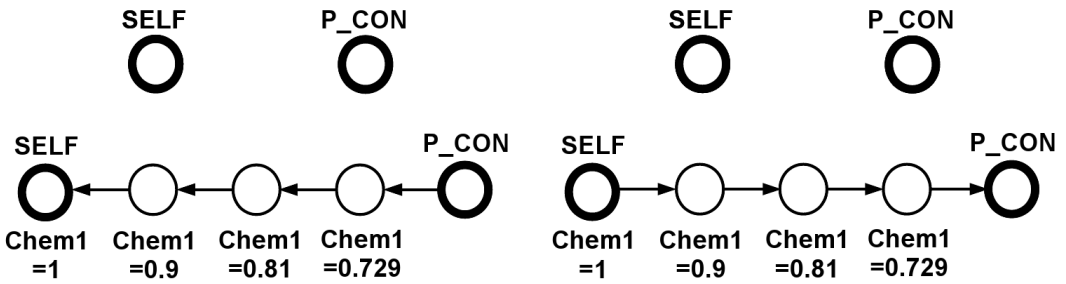
Action	**Parameters**	**Description**
ADD_N1_I **ADD_N1_O**	CHEM, QTY, FUNC	Creates QTY intermediary nodes of function FUNC in sequence from P_CON/E_CON to SELF (ADD_N1_I) or from SELF to P_CON/E_CON (ADD_N1_O), with chemical levels determined by CHEM (fig. 4.3).
ADD_N2_I **ADD_N2_O**	CHEM, QTY, FUNC	Create an intermediary node of function FUNC between SELF and each set of QTY input (ADD_N2_I) or output (ADD_N2_O) nodes, with chemical levels determined by CHEM (fig. 4.4).
DEL_N		Remove node
ADD_C_I **ADD_C_O**	WEIGHT	Add input (ADD_C_I) or output (ADD_C_O) connection with weight specified by WEIGHT
DEL_C_I **DEL_C_O**		Delete input (DEL_C_I) or output (DEL_C_O) connection

Table 4.4. Actions in the genome. SELF refers to the node for which actions are being considered, E_CON (existing connection) refers to a node that shares an existing direct connection with SELF, and P_CON (potential connection) refers to a node that does not share a direct connection with SELF. 'I' stands for 'input' and 'O' stands for 'output'. Unlike in chapter 3, the genome can now develop new nodes in addition to new connections.

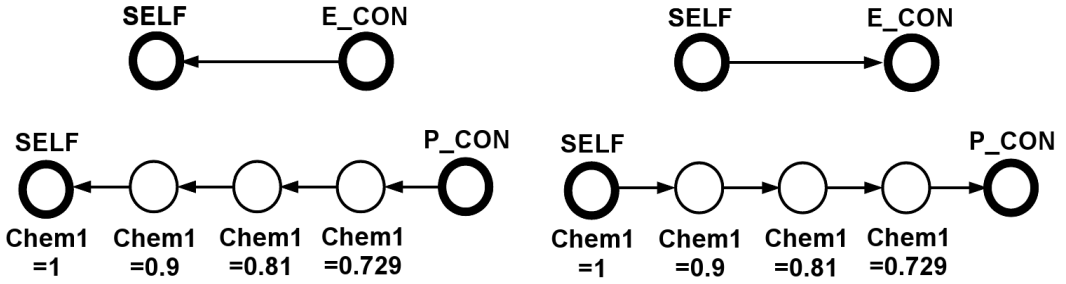
and z respectively. These particular values would later become necessary for aligning role nodes with inhib2 nodes in fact circuit development (section 4.4)[1]. Fact nodes, inhibitory nodes and mediator nodes may also be assigned chemical levels, but these nodes are only produced through development and therefore do not have chemical levels prescribed to them. Instead their chemical levels are determined through inheritance. When a condition in the genome tests for equality of chemical levels (e.g. $SELF.Chem1 = 0.5$), approximate equality is tested for within a margin of error of 0.02, enough to distinguish between role nodes by their chemical levels.

Throughout development, SHRUTI nodes (neurons) and their connections can be tested according to the conditions listed in table 4.3 and modified according to the actions listed

[1]Fact development begins at a predicate's collector ($Chem2 = 1$), from which the genome produces a fact node and then one inhib2 node for each of up to three predicate arguments. Each node inherits 90% of its predecessor's chemicals so that the fact node has a chemical 2 value of 0.9, and the three inhib2 nodes have chemical 2 values of 0.81, 0.729 and 0.6561.

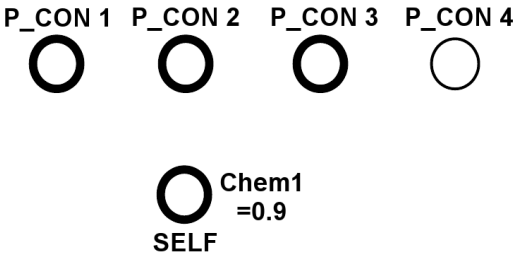


(a) ADD_N1_I (P_CON) (b) ADD_N1_O (P_CON)

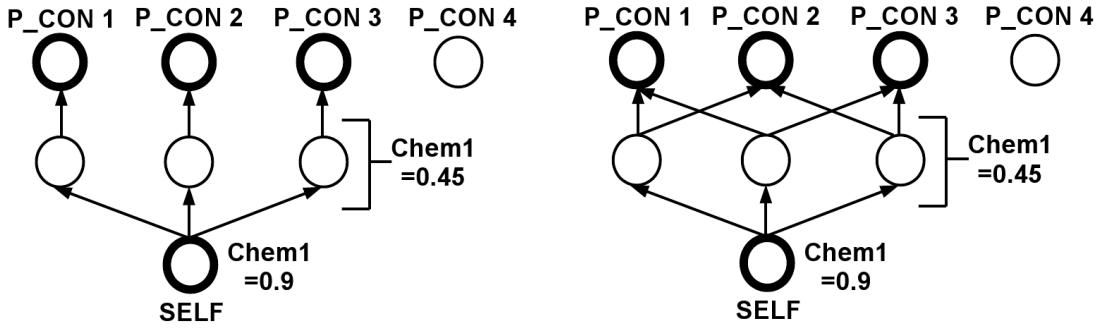


(c) ADD_N1_I (E_CON) (d) ADD_N1_O (E_CON)

Figure 4.3. Creating intermediary nodes between SELF and P_CON/E_CON using ADD_N1 with parameters QTY = 3 and CHEM = 0.9. Node formation begins at SELF and each successive new node inherits a percentage of its predecessor's chemicals determined by CHEM. E_CON becomes P_CON because it no longer shares a direct connection with SELF.



(a) Before



(b) QTY = 1 (c) QTY = 2

Figure 4.4. Creating intermediary output nodes using ADD_N2_O with CHEM = 0.5 and different values for the QTY (quantity) parameter. Three node pairs (SELF, P_CON 1), (SELF, P_CON 2) and (SELF, P_CON 3) satisfy some rule that triggers ADD_N2_O, whereas (SELF, P_CON 4) does not. An intermediary node is added between SELF and each set of QTY P_CON nodes that satisfies the rule. ADD_N2_I works in a similar way but with arrows reversed. Bold circles indicate nodes that satisfy the conditions for development to take place.

in table 4.4. As before, SELF refers to the node for which actions are being considered. Some conditions and actions now have input and output alternatives, as the genome no longer considers potential or existing inputs (P_INPUT and E_INPUT) only, but instead considers potential and existing connections (P_CON and E_CON respectively) that can either be inputs or outputs to SELF. For example, the condition WEIGHT from the previous model has been replaced with the conditions WEIGHT_I and WEIGHT_O for testing the weight of input and output connections to SELF respectively. Although the genomes in sections 4.3 and 4.4 only require some conditions and actions to have input and output alternatives, these were introduced for all appropriate conditions and actions for completeness.

The actions ADD_N1_I and ADD_N1_O create a set of intermediary nodes in series between SELF and P_CON or E_CON as shown in fig. 4.3. Node formation begins at SELF and each successive node created inherits a portion of its predecessor's chemicals as determined by the CHEM parameter.

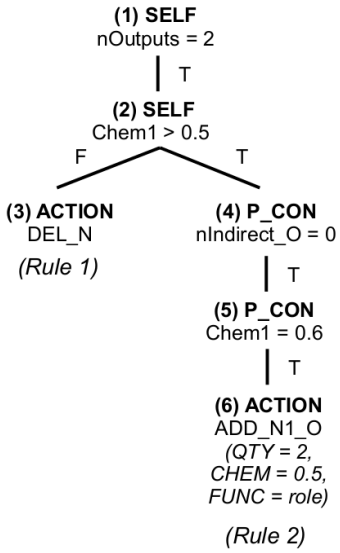
ADD_N2_I and ADD_N2_O create an intermediary input or output node respectively for every set of QTY potential input or output nodes that meet any necessary conditions pertaining to P_CON, as shown in figure 4.4. Such behaviour was necessary for the formation of mediators (section 4.3.1). If an intermediary node already exists between SELF and a set of P_CON nodes, no duplicate is created. It is assumed that the information as to whether or not two nodes share an intermediary node could be passed between neurons using neuromodulators. The same is assumed when evaluating the conditions nIndirect_I and nIndirect_O.

A network is produced from the genome according to the principles introduced in section 3.2.2. Fig. 4.5 on the next page presents an example of network development using some of the new conditions and actions. The first rule in fig. 4.5a deletes a SHRUTI node (action 3) which has two outputs (condition 1 = True) and 0.5 or less of chemical 1 (condition 2 = False). In fig.4.5b, node H is the only node for which both of these conditions are satisfied, so it is removed (fig. 4.5c). The second rule in fig. 4.5a produces a sequence of two intermediary role nodes (action 6) between SELF and potential connection P_CON when conditions specified for SELF and P_CON are met. Only node G satisfies the conditions for SELF because it has two existing outputs (condition 1 = True) and over 0.5 of chemical 1 (condition 2 = True). Only node A meets the conditions for P_CON because it shares no intermediary nodes with SELF/node G (condition 4 = True) and it has 0.6 of chemical 1 (condition 5 = True). Therefore according to action 6, two intermediary role nodes I and J are produced to propagate activation from node G to node A (fig. 4.5d). Node I is produced before node J, and each inherits 0.5 of its predecessor's chemical 1 level according to the parameter CHEM = 0.5.

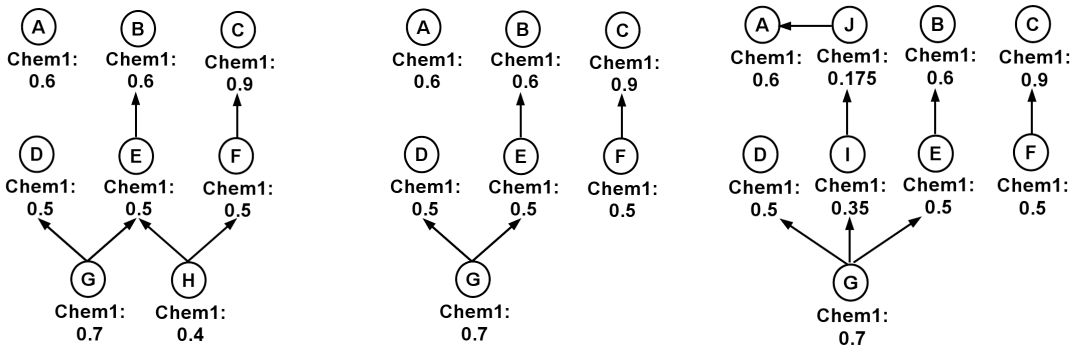
This section and section 4.1 introduced the learning algorithm and developmental model necessary for producing mediator and fact structures. How these methods were used to produce each structure specifically is explained in the following sections, 4.3 and 4.4.

4.3. Mediator structures

This section investigates a genotypic representation of SHRUTI's mediator structures and the evolvability of such genomes. A diagram of SHRUTI mediator structures can be found in fig. 2.9 on page 44. In the original SHRUTI literature, no learning algorithm suitable for mediator structures was proposed, so one is proposed in section 4.3.1 before introducing and testing the genome for mediators in sections 4.3.2 and 4.3.3. Section 4.3.4 explains necessary changes to the evolutionary algorithm in order to search for mediator genomes, and section 4.3.5 presents the results. Although the evolutionary search did not discover genomes capable of producing mediator structures, an investigation of the fitness landscape in section 4.3.6 was able to identify why so that future research can concentrate



(a) The genome. Terminal nodes are actions, all other nodes are conditions, and each path from the root of the tree to a terminal node is a different developmental rule. SELF refers to a neuron for which actions are being considered, and P_CON refers to a potential neuron from which to produce an input or output connection.



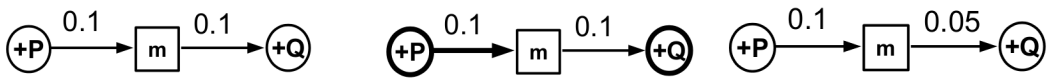
(b) Before development (c) Rule 1 is applied (d) Rule 2 is applied

Figure 4.5. An example of genome development using some of the new conditions and actions introduced in this chapter. For a further understanding of how SHRUTI genomes in general lead to complete SHRUTI network structures, see section 3.2.2. Only neuron H satisfies the conditions for rule 1, and so it is removed (c). Rule 2 then produces a sequence of two intermediary nodes between SELF and P_CON when SELF = neuron G and P_CON = neuron A, as these are the only two neurons that meet the conditions of the rule (d). A more complete explanation of the rules and actions used can be found in the text.

on improving evolvability. Due to the brittle nature of SHRUTI networks and the genomes required to represent them, the fitness landscape contained insufficient information to support an evolutionary search in identifying individual components necessary for the construction of mediator structures.

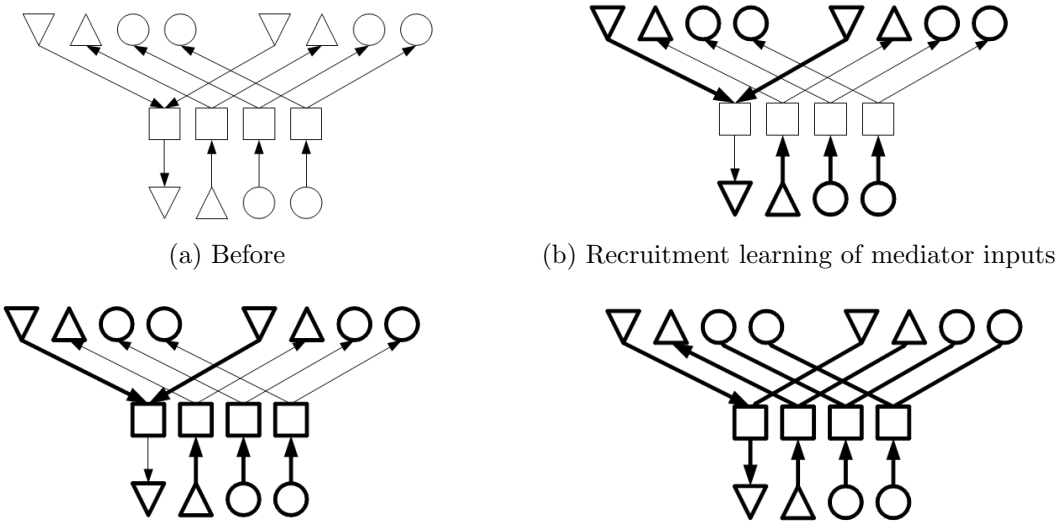
4.3.1. Learning mediator structures

Causal Hebbian learning requires that a node and its input must be active in order for the connection between them to strengthen (page 49). The difficulty in applying causal Hebbian learning to mediator structures is that a mediator structure separates connections between predicate clusters so that there is no longer a set of direct connections between the nodes for two predicates P and Q. Instead, connections exist from the nodes of P to the mediator nodes between them and from those mediators to the nodes of Q. Because the mediator nodes cannot be activated by observation in the way nodes for P and Q are when they are observed, nodes at P and Q are activated with no activations at the mediator nodes so that connections from the mediator nodes weaken according to the rules of causal Hebbian learning. For example, consider the collectors $+P$ and $+Q$ for each predicate and a mediator node m between them (fig. 4.6a). When $+P$ and $+Q$ co-activate, no activation



(a) (b) (c)

Figure 4.6. According to causal Hebbian learning, a direct connection between $+P$ and $+Q$ would strengthen when they coactivate. However mediator nodes interrupt the causal Hebbian learning process so that $+Q$ is active with no activation from its input. Therefore, the input connection to $+Q$ weakens. Bold circles represent active nodes.



(a) Before (b) Recruitment learning of mediator inputs

(c) Activation of recruited mediator nodes (d) Hebbian learning of mediator outputs

Figure 4.7. Learning of mediator structures. Active nodes and connections being strengthened are displayed in bold. Mediators are recruited through recruitment learning, in which any inputs participating in the activation of a mediator node are strengthened through LTP. Connections from mediators to predicate nodes are then strengthened through causal Hebbian learning.

NoNeg5

Relations	Facts
$P(x,y) \wedge Q(x,y) \rightarrow R(x,y)$	$P(a,b)$
$R(x,y) \rightarrow S(y)$	$Q(a,b)$
	$R(c,d)$

NoNeg6

Relations	Facts
$P(x,y,z) \rightarrow Q(x,y)$	$P(a,b,c)$
$Q(x,y) \wedge R(x,y) \rightarrow T(x,y)$	$R(a,b)$
$S(x,y) \rightarrow T(x,y)$	$Q(d,e)$
	$R(d,e)$
	$S(f,g)$

NoNeg7

Relations	Facts
$P(x,y) \rightarrow R(x,y)$	$P(a,b)$
$Q(x,y) \wedge R(x,y) \rightarrow T(x,y)$	$P(c,d)$
$R(x,y) \wedge S(x) \rightarrow U(x)$	$Q(a,b)$
	$S(c)$

NoNeg8

Relations	Facts
$P(x,y,z) \rightarrow R(y,z)$	$P(a,b,c)$
$Q(y) \rightarrow S(y)$	$P(d,e,f)$
$R(y,z) \wedge S(y) \rightarrow U(y)$	$Q(b)$
$S(y) \wedge T(x,y) \rightarrow V(y)$	$Q(h)$
	$S(e)$
	$T(a,b)$
	$T(g,h)$

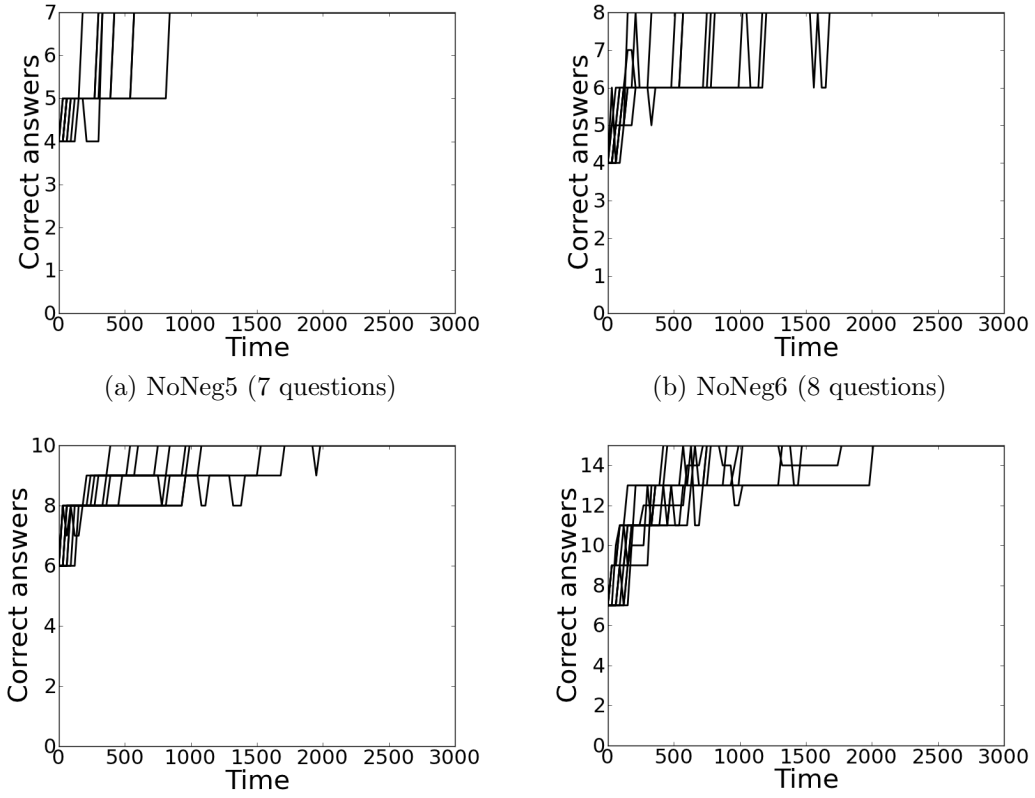
Figure 4.8. Logic programs containing conjunctive relations.

propagates from $+P$ to $+Q$ because the connection between $+P$ and m is too weak to activate m (fig. 4.6b). Therefore $+Q$ is activated by observation with no pre-synaptic activation from its input (m) and the connection from m to $+Q$ weakens.

The proposed solution is to use recruitment learning to recruit mediator nodes by strengthening their inputs through Long-Term Potentiation (fig. 4.7b), and to use causal Hebbian learning to strengthen mediator outputs (fig. 4.7d). Predicates are observed and corresponding predicate nodes become active. Outputs from these nodes then cross the recruitment thresholds of mediator inputs, strengthening these connections (fig. 4.7b). Eventually these connections gain enough strength to activate the mediator nodes (fig. 4.7c) and the remaining connections from mediator nodes to other predicate nodes can be strengthened using causal Hebbian learning, as before (fig. 4.7d).

In chapter 3, the prerequisite for SHRUTI to learn all of its target relations was a network of nodes fully interconnected by function. For SHRUTI networks involving mediator structures, the prerequisite structure contains a mediator structure for every possible relation of up to and including two antecedents. In other words, a mediator structure is created for every conjunctive and non-conjunctive relation. Only nodes of the same function share mediator nodes and connection weights are initially low.

The learning algorithm was tested on the logic programs shown in fig. 4.8, using probabilistic event sequences and B question sets defined in appendix A. Logic programs including negated predicates were excluded from these and all further experiments, having demonstrated the difficulties that SHRUTI has in learning them in chapter 3. Where evidence for a conjunction is presented in an observation sequence, for example $P(x,y), Q(x,y)$ before $R(x,y)$ to support $P(x,y) \wedge Q(x,y) \rightarrow R(x,y)$, instances of $P(x,y)$ and $Q(x,y)$ occurring individually and without $R(x,y)$ following must also be presented in order to sufficiently weaken any relational structures that represent $P(x,y) \rightarrow R(x,y)$ and $Q(x,y) \rightarrow R(x,y)$. This increased the time it took to learn a relation compared with chapter 3, and therefore longer event sequences were necessary for all relations to be learned. Because activation



(a) NoNeg5 (7 questions)

(b) NoNeg6 (8 questions)

(c) NoNeg7 (10 questions)

(d) NoNeg8 (15 questions)

Figure 4.9. The number of correct answers to questions over time for each NoNeg data set containing conjunctive relations. 10 trials were performed for each logic program, generating a new event sequence each time. Each network learns to answer all of its test questions correctly.

now needs to propagate via mediator structures, activation takes longer to propagate between the collectors of two predicates when the antecedent activates and between enablers and role nodes when the consequent activates. Therefore a longer window of synchrony than was used in chapter 3 was required to enable relations to be learned. A time window of 5 observations was found to be suitable. Fig. 4.9 shows the results of learning each logic program in fig. 4.8 from the probabilistic event sequences. Each network learns to answer all test questions correctly.

One disadvantage of the updated learning model is that each time a predicate's nodes are activated, all mediator nodes they output to will be recruited regardless of whether or not the outputs of those mediators are also activated (fig. 4.10), leading to the learning of many unwanted connections. However, it was hoped that evolution might find genomes that would reduce the number of redundant connections such as these. Even if these connections are not removed, strengthening them may serve to prime them for future relations that may

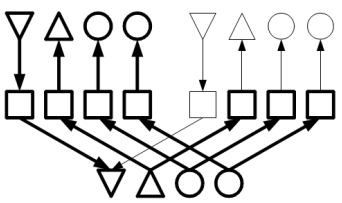


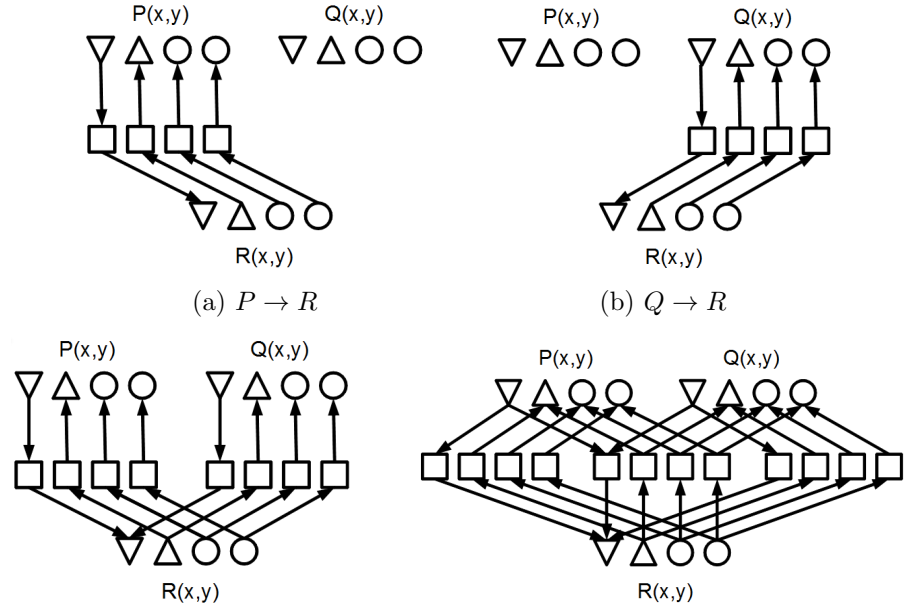
Figure 4.10. Recruitment of unwanted mediator nodes (right)

be discovered should evidence of new relations ever be presented. Either way, although there are still improvements that could be made to the new learning algorithm, it is sufficient to learn all logic programs in figure 4.8 and demonstrate the arguments this thesis makes about the development and evolution of SHRUTI networks.

4.3.2. Developing mediator structures

In chapter 3, the bare minimum requirement for SHRUTI to learn all target relations was a network of nodes fully interconnected by function. With mediator structures now included in the SHRUTI model, the bare minimum requirement was a mediator structure for every possible conjunctive and non-conjunctive relation. A genome for producing such a network is introduced in this section, and is labelled G5. However, fully interconnected networks become very expensive as the number of predicates increases, and a better approach would be to restrict the number of connections whilst ensuring that those necessary for learning target relations are constructed at some point. The best strategy for this as demonstrated in chapter 3 was to limit the development of connections to active neurons. A genome that performs these restrictions for mediator structures is also introduced in this section and is labelled G6. The structures of both G5 and G6 can be found later in figures 4.13 and 4.14 on pages 132 and 133, and detailed descriptions of the rules, conditions and actions used to implement them are also provided later. First, the following paragraph describes the overall developmental process.

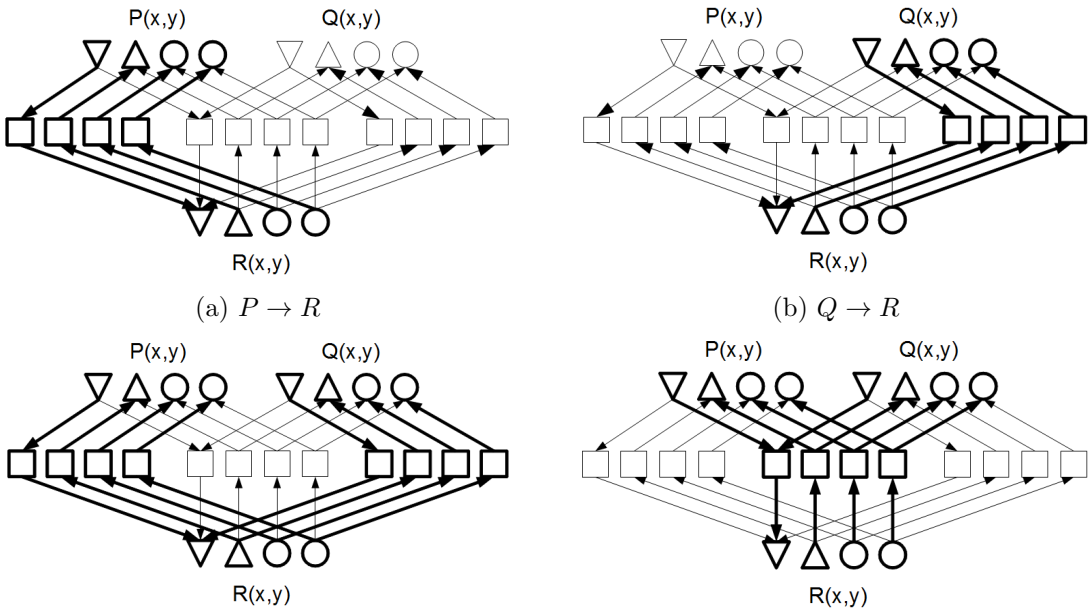
The development of mediator structures occurs in two stages. For any set of two antecedents P and Q and one consequent R, both genomes begin by producing structures



(a) $P \rightarrow R$ (b) $Q \rightarrow R$

(c) $P \wedge Q \rightarrow R$ (stage 1) (d) $P \wedge Q \rightarrow R$ (stage 2)

Figure 4.11. Development of relations and mediator structures. The development of a conjunctive relation occurs in two stages. First, mediators for individual non-conjunctive relations must develop (stage 1), before the conjunctive mediator structure develops (stage 2). The learning algorithm will adjust connection weights to reflect which of the developed relations are observed in the event sequence (fig. 4.12).



(a) $P \rightarrow R$ (b) $Q \rightarrow R$

(c) $P \vee Q \rightarrow R$ (d) $P \wedge Q \rightarrow R$

Figure 4.12. Learning relations after development of mediator structures. Bold shapes represent active nodes and bold lines represent connections gaining strength to reflect observed evidence for particular relations.

for the individual non-conjunctive relations $P \rightarrow R$ and $Q \rightarrow R$ (stage 1, fig.4.11c) before producing the structure for the conjunctive relation $P \wedge Q \rightarrow R$ (stage 2, fig. 4.11d). Note that the construction of stage 1 is equivalent to the behaviour of group 1 genomes from section 3.4.2 as they produce structures for non-conjunctive relations by producing connections between nodes of the same function. The development of each structure is immediate in the case of G5, but in the case of G6 this development is triggered by activation. If the evidence for $P \rightarrow Q$ and $Q \rightarrow R$ are observed separately, G6 develops the corresponding structures separately (figs. 4.11a and 4.11b). If they are observed together in the form of evidence for $P \wedge Q \rightarrow R$, they are developed together (fig.4.11c). Either way, as soon as both non-conjunctive structures exist and evidence for $P \wedge Q \rightarrow R$ is observed again, the conjunctive mediator structure is formed (fig.4.11d). The three relations that can be learned from the structure that develops are $P \rightarrow R$, $Q \rightarrow R$ and $P \wedge Q \rightarrow R$. Not all of these relations may be true in the target logic program, but the learning algorithm will adjust the connection weights of the mediator structures accordingly (fig. 4.12). Of course, both $P \rightarrow R$ and $Q \rightarrow R$ imply $P \wedge Q \rightarrow R$ so that if either is true in a logic program, the conjunctive relation is also. However the learning algorithm cannot make such conclusions automatically and the conjunctive relation will only be learned if evidence for it is observed in the form of P and Q occurring together shortly before an observation of R. It is also important to note that $P \wedge Q \rightarrow R$ does not imply either $P \rightarrow R$ or $Q \rightarrow R$ and therefore that $P \wedge Q \rightarrow R$ must be representable without the others. Although the structures for all three are constructed by G6 when the conjunction is observed, they are not necessarily all learned because connections representing $P \rightarrow R$ and $Q \rightarrow R$ will weaken if P and Q are observed independently without R.

The following paragraphs elaborate on the choice of rules, conditions and actions employed to enable the development of mediator structures. Genomes G5 and G6 are shown in

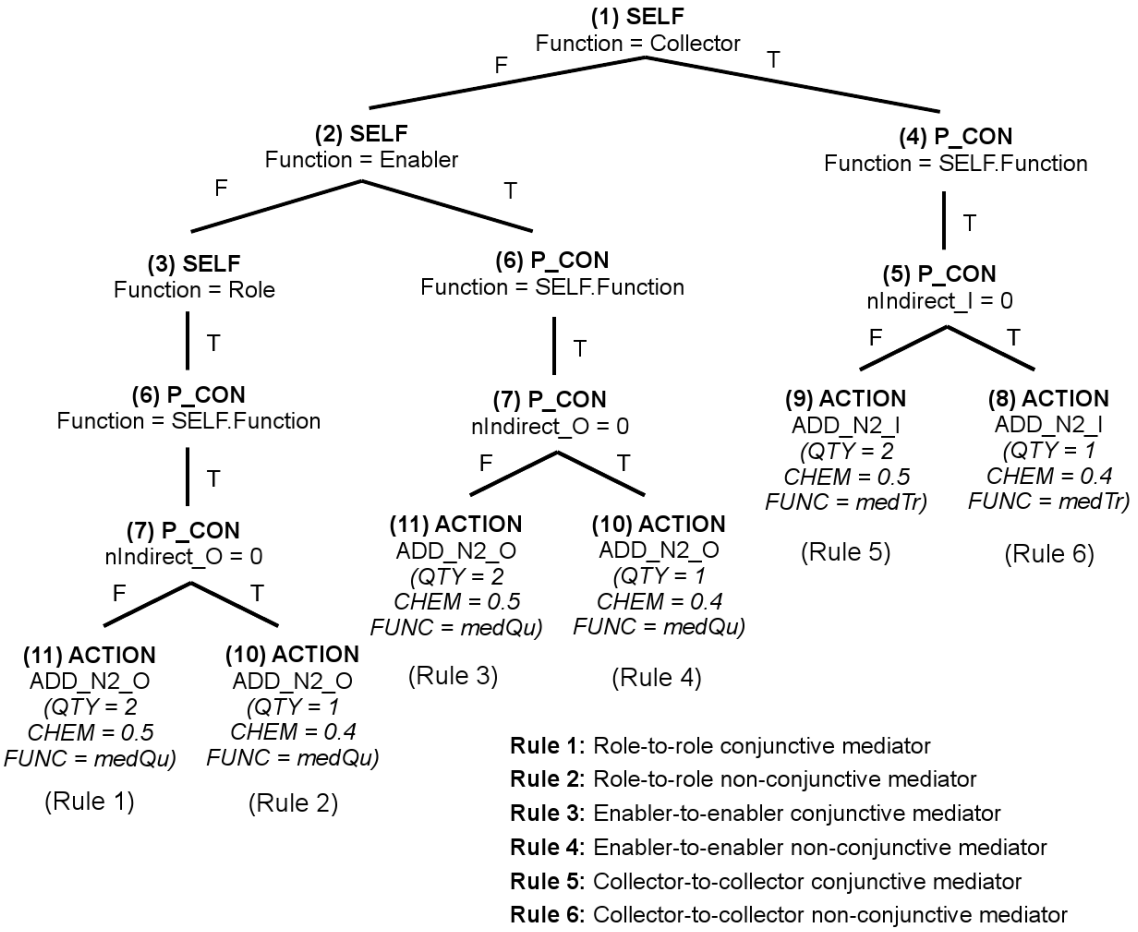


Figure 4.13. Genome G5. A mediator structure is produced for every possible relation of up to and including two antecedents. The choice and meaning of the rules, conditions and actions used are explained in the text.

figs 4.13 and 4.14. These new genomes differ from those for producing non-conjunctive relations in chapter 3 in a number of ways, discussed below. As described in section 4.2, a node labelled SELF in the genome is the node for which actions are being considered, P_CON refers to a potential node with which to produce a connection and E_CON refers to an existing connection.

Whereas in chapter 3, all inter-function connections (e.g. enabler-to-enabler) could be created using the same action, mediator nodes between collectors (medTr) must be produced by a different action from mediator nodes between enablers and role nodes (medQu). This is because medTr nodes have different functionality from medQu nodes and therefore they must be produced using different actions. The two node functions differ in that medTr nodes have multiple inputs and a single output, and medQu nodes have a single input and multiple outputs. Therefore the former must be produced using ADD_N2_I and the latter using ADD_N2_O (see table 4.4 and figure 4.4 for an explanation of these actions). The two node addition actions are executed with $QTY = 1$ for individual relations and $QTY = 2$ for conjunctive relations. Different values are also specified for the parameter $CHEM$, which encodes the proportion of the parent node's chemicals to be inherited by each child. Chemical values reveal a node's position relative to others so that they may be distinguished. Therefore mediators for non-conjunctive relations can be distinguished

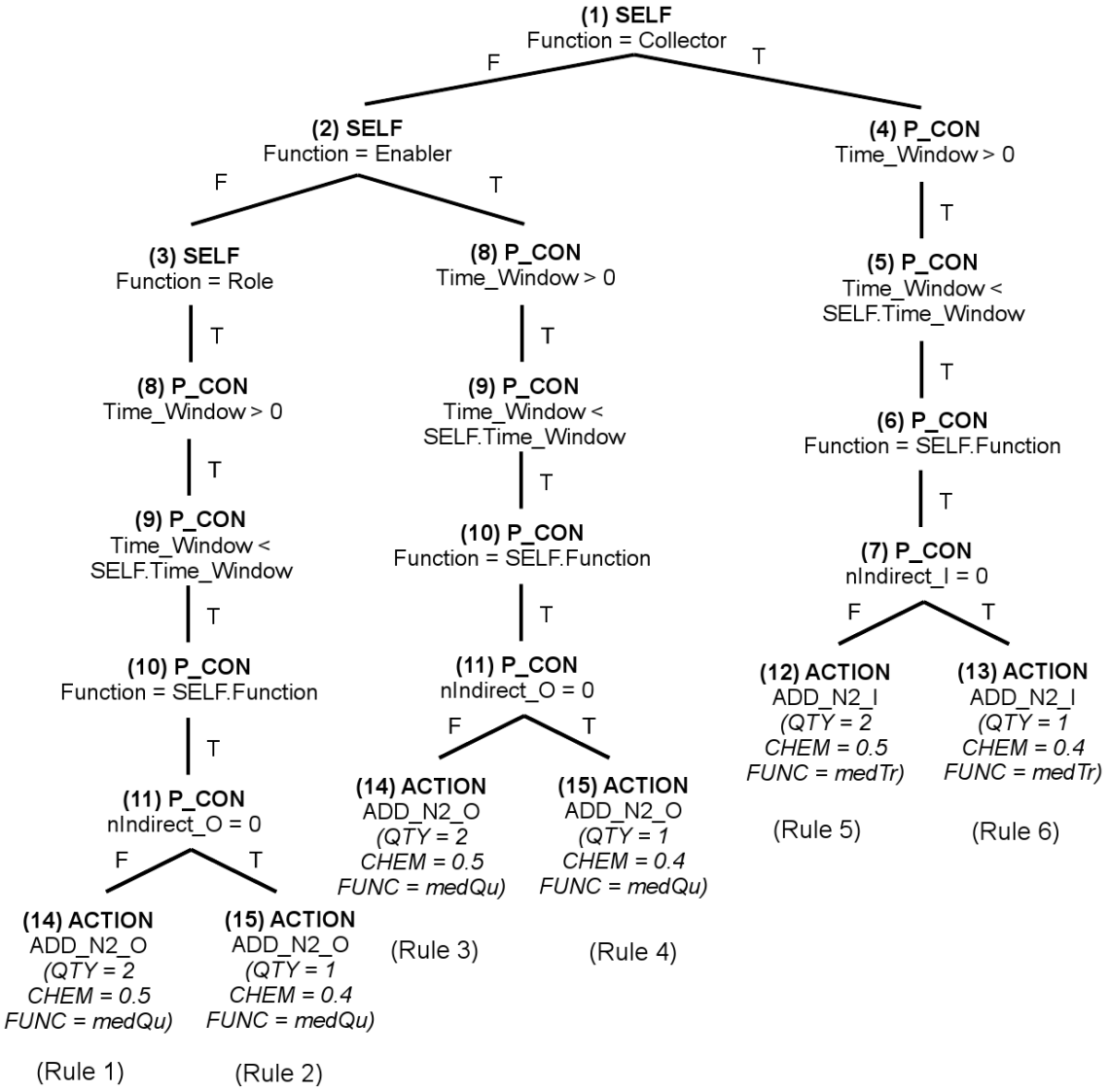


Figure 4.14. Genome G6. A mediator structure is produced for every relation for which evidence is observed in the event sequence used for training. The choice and meaning of the rules, conditions and actions used are explained in the text.

from those for conjunctive relations based on different chemical amounts inherited from the same parent. For example, when producing mediators between enablers, which have a value of 1 for chemical 1, a conjunctive mediator will be produced with a value of 0.5 for chemical 1 and a non-conjunctive mediator will be assigned a value of 0.4 for chemical 1.

The ordering of structure development (individual non-conjunctive structures before conjunctive structures) and therefore which of these to produce first is enforced by the condition $nIndirect\_I = 0$ for collectors and $nIndirect\_O = 0$ for enablers and role nodes. As explained in table 4.3, these conditions test for the number of indirect inputs or outputs (paths separated by one node) between two nodes and can therefore be used to count the number of mediator nodes between them. $nIndirect\_I = 0$ or $nIndirect\_O = 0$ evaluating 'true' tells the genome that a non-conjunctive mediator node does not exist and must be constructed (figs 4.11a, 4.11b and 4.11c). If 'false', a non-conjunctive mediator node must exist and the genome can construct the conjunctive mediator node (fig. 4.11d).

Function equality can no longer be enforced by a single condition $P\_CON.Function =$

SELF.Function (the current node and its potential connection perform the same function) as was the case in the chapter 3 (fig. 3.9, page 82). This is partly because collector nodes must be treated differently from enabler and role nodes in that different mediator nodes are formed, and therefore it is important to distinguish node function. Furthermore, collectors, enablers and role nodes are no longer the only node functions that may be tested for by conditions in the genome, and therefore function equality must be more specific in order to avoid the construction of unwanted connections between two nodes of other functions, for example medTr-to-medTr or medQu-to-medQu.

G6 also contains conditions on 'time window' (the remaining time before an active node's window of synchrony expires) for each node function in order to restrict the development of mediator nodes to active nodes. The remaining time window for P_CON must be greater than 0 but less than that of SELF, ensuring that both SELF and P_CON have remaining time windows above 0 and are therefore both active. Forcing the time window for P_CON to be less than that of SELF also ensures that the antecedent is active before the consequent so that mediators are only created for $P \rightarrow Q$ or $Q \rightarrow P$, but not both.

With so many conditions, genomes G5 and G6 are much more complex than the genomes presented in chapter 3. The ease of discovering genomes through evolution in chapter 3 was largely owed to the fact that function equality could be enforced by only one condition, but now it must be enforced by five in precise combination. This made it very difficult for mediator genomes to be rediscovered through evolution, as is demonstrated later in sections 4.3.5 and 4.3.6.

4.3.3. Testing development

Fig. 4.15 shows the results of developing networks for each logic program using genome G6, 10 probabilistic event sequences and the B question sets (appendix A). As with the learning experiments, a time window of 5 was used. In each case, the developed network is eventually able to answer all of its test questions correctly. The results of developing networks using G5 are equivalent to the results presented in fig. 4.9 on page 129 because G5 produces a structure for every possible relation and it was on such a network that the learning algorithm was tested. Each logic program on which the networks were trained vary in size, showing that like the genomes in chapter 3.2, the genomes are scalable.

Tables 4.5 and 4.6 present the statistics of networks developed for each logic program using both genomes. As expected, G5 produces the largest networks because it produces a structure for every possible relation of two antecedents or less. The number of weight updates is also considerably higher than that yielded by G6, thus demonstrating that restricting development by activation (G6) is more efficient than not (G5).

The average logic program from NoNeg1-4 (fig. 3.1) and Neg1-4 (fig. 3.3) in chapter 3 contained 4.625 predicates, whereas the average logic program in NoNeg5-8 (fig. 4.9), used in these experiments, has an average of 4.25 predicates. Genomes in both chapters therefore developed networks of similar sizes. However both G5 and G6 produce many

G5: A mediator structure is produced for every possible relation.

Program	#Relations	#Predicates	#Nodes	#Connections	#Updates
NoNeg5	5	3	440	1111	5411
NoNeg6	6	4	1173	3157	19567
NoNeg7	7	4	1389	3786	13742
NoNeg8	8	6	2106	5818	28541

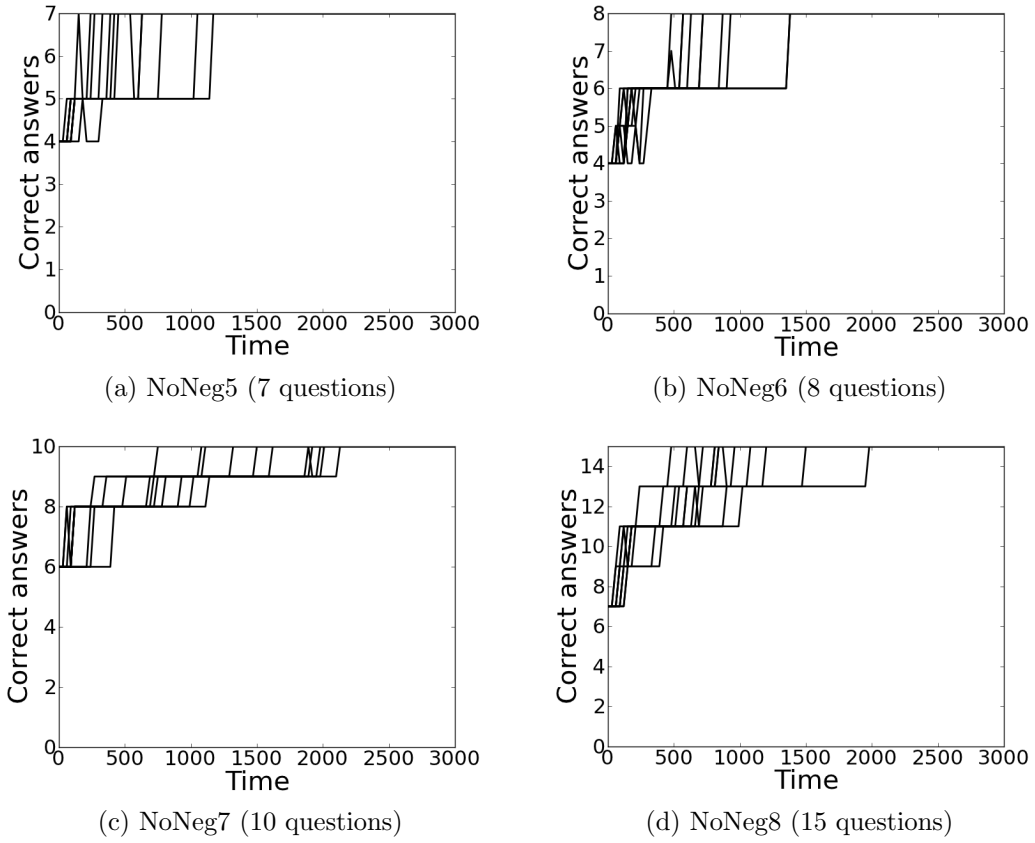
G6: The development of mediator structures is restricted by activation.

Program	#Relations	#Predicates	#Nodes	#Connections	#Updates
NoNeg5	5	3	76	113	1050
NoNeg6	6	4	130	214	3165
NoNeg7	7	4	103	155	1665
NoNeg8	8	6	146	238	3404

Table 4.5. Statistics for each genome developing networks for each logic program using fixed event sequences. All test questions are answered correctly in each case. Each logic program and developed network is a different size, despite the fact that the same fixed-size genome is used in each case. This shows that the genome is scalable.

Program	#Nodes	#Connections	#Updates
G5	1277	3468	16815.25
G6	113.75	180	2321

Table 4.6. Average statistics for both genomes



(a) NoNeg5 (7 questions)

(b) NoNeg6 (8 questions)

(c) NoNeg7 (10 questions)

(d) NoNeg8 (15 questions)

Figure 4.15. Developing SHRUTI networks using genome G6. The graphs show the number of correct answers to questions over time for each NoNeg data set containing conjunctive relations. 10 trials were performed for each logic program, generating a new event sequence for each trial. Each network developed so that it could answer all of its test questions correctly.

more connections than their equivalents in chapter 3. G5 is equivalent to genome G1 in that it produces a relational structure for every possible relation. The average number of connections in G1 was 197.5, but 3468 for G5. The equivalent of G6 in chapter 3 is G3, which only produces structures for observed relations. The average number of connections in G3 was 78.25, and G6 produces an average of 180. Both G5 and G6 therefore do more work than their equivalents because in addition to producing a structure for every non-conjunctive relation, one for every possible conjunctive one of two antecedents must also be produced. Furthermore, the number of connections produced for any individual relation is even greater since antecedents and consequents are separated by mediator nodes.

As was the case in the previous chapter, in order for any relation to be learned, the structure representing it must exist to be learned in the first place. This is a consequence of the localist nature of SHRUTI. In order to guarantee that a desired relation exists, the genome must either produce a structure for every possible relation (G5) or construct the desired relation when it is needed (G6), that is when evidence for it is observed. The former is inefficient and lacks biological plausibility in that it suggests that our brain contains a representation for every possible relation we could learn. Although the latter has some biological plausibility in that there is evidence for the activity-dependent development of neural structures [33], it still lacks biological plausibility in that some conditions in the genome match some conditions for learning, thus making aspects of learning redundant. As argued in chapter 3, distributed encodings may provide a solution to this dilemma as they could enable development to support learning without replacing it, by improving factors such as connectivity and storage capacity akin to the information processing theory of development [69, 100].

Nonetheless, regardless of what triggers the development of SHRUTI mediator structures, the results show that they have been successfully represented in a scalable genome, thus supporting the claim of SHRUTI's developers that SHRUTI structures can be represented by biologically plausible indirect encodings.

4.3.4. Method for evolving mediator structures

Now that genomes for developing mediator structures had been produced, the next goal was to explore the possibility of discovering similar genomes through evolution in order to determine the evolvability of SHRUTI networks. This section explains how these evolutionary experiments were performed. Evolution was attempted in much the same way as in chapter 3, although some parameters were modified and constraints were set on network size and development time. Furthermore, following some preliminary tests it became necessary to modify the error function to weight group 2 and 3 networks more negatively.

Before continuing, some clarification of terms is necessary. In chapter 3 (section 3.4.2), the term 'group 1 genome' referred to a genome that yielded zero-error by correctly developing connections between nodes of the same function. However, this is only sufficient for non-conjunctive relations and now that conjunctive structures must also be discovered, a group 1, zero-error genome from chapter 3 will not yield zero-error for questions dependent on

Logic program	Time limit (seconds)	Maximum size (nodes)
NoNeg5	60	1000
NoNeg6	120	2000
NoNeg7	120	2000
NoNeg8	180	3000

Table 4.7. Constraints on network size and development time when evolving networks containing mediator structures. Networks which develop beyond these limits are penalised to the lowest rank.

conjunctive relations in this chapter. The term '*group 0 genome*' is introduced to refer to genomes such as G5 and G6 that successfully represent mediator structures so that all questions on both types of relation can be answered correctly. In other words, group 1 genomes are sufficient to produce zero-error networks for logic programs that don't include conjunctive relations, and group 0 genomes are sufficient to produce zero-error networks for logic programs that do. Note also that as explained in section 4.3.2, the development of networks produced by G5 and G6 occurs in two stages. Stage 1 is equivalent in behaviour to group 1 genomes that produce non-conjunctive relational structures, and stage 2 is equivalent to group 0 genomes that can produce conjunctive relational structures. As before, group 2 genomes are those that only produce connections between enablers and collectors, and group 3 genomes are those that do not produce connections at all and therefore answer all questions as [0,0] (unknown).

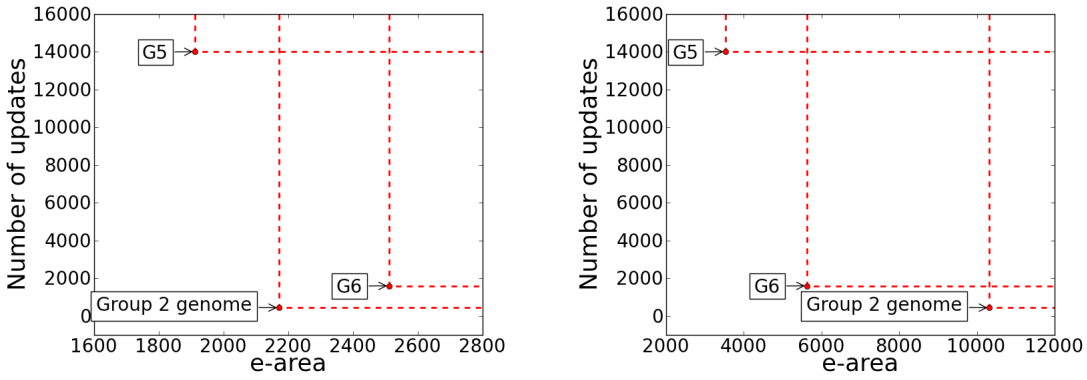
Now that the definition of each group of genome has been clarified, the remainder of this section explains the method used to evolve mediator genomes. As explained in section 4.3.1, it takes longer to learn relations from NoNeg5-8 than it did to learn logic programs in chapter 3. Around 500 observations, including time steps at which no observations occur, were necessary for learning the largest logic program (NoNeg8) and therefore this was the number of event observations chosen for evaluating genomes evolved for all networks. The genome size was also increased. G6 contains 15 conditions and actions, so to allow similar genomes to be discovered and to allow some flexibility in the structure, genomes were evolved with a size of 25 conditions or actions. As in section 3.3.3, the A question set for each logic program (appendix A) was used to evaluate fitness.

A limit was placed on development time and the number of nodes that could be produced in order to avoid producing networks that grew so large that measuring fitness became intractable. Experiments were performed on a 3.07 GHz Intel Core i7 processor with 6 GB of RAM, and networks were implemented using Python 2.7. With a mediator structure for every possible conjunctive and non-conjunctive relation, evaluating fully interconnected networks over a number of observations sufficient to learn all relations was very expensive, taking up to two minutes per network for NoNeg8, the largest network. Evaluating more restrictive genomes such as G6 was much cheaper, but in chapter 3 genomes that restrict development by activation were usually descended from genomes that produced networks containing a structure for every possible relation, and therefore it was important to set restrictions on time and network size that enabled G5 or similar genomes to be discovered. Each logic program was constrained by a time limit and maximum network size, which when exceeded, would cause the genome to be penalised to the lowest rank in the population. The limits for each logic program are presented in table 4.7 and in each case

are more than sufficient to develop the corresponding zero-error networks using G5.

Preliminary experiments attempting to produce networks for NoNeg8 were performed using the configuration described so far. From observing the genomes produced it became clear that the error function defined in section 3.3.4 was no longer suitable for this task. Using the old measure, G5 yields an e-area of 1911 with 14007 weight updates and G6 yields an e-area of 2511 with 1639 weight updates, but a group 2 genome (a genome that always answers 'true' for any predicate) discovered during these preliminary experiments yielded an e-area of 2171 with 648 weight updates. A plot comparing these three genomes can be found in fig. 4.16a. Among the three genomes, G5 and the group 2 genome are non-dominated. However, G6 is dominated by the group 2 genome and therefore similar genomes may easily be lost from the population during evolution.

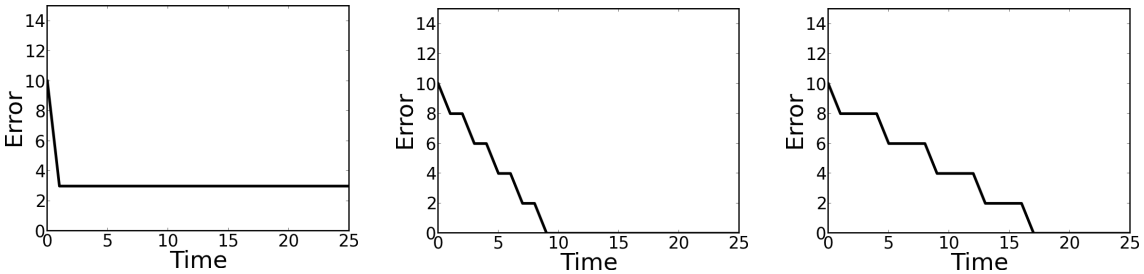
e-area is proportional to time. In chapter 3, the time window for synchrony was set to 2. In these experiments, a time window of 5 was used to account for the time it took to propagate activation across mediator structures. The shorter time window used before meant that development using activity-dependent genomes was faster because the observations that triggered development occurred over a shorter period of time, and therefore e-area was smaller (fig. 4.17b). Now that experiments were performed with a time window of 5, development took longer and e-area was larger (fig.4.17c). In both this chapter and the



(a) Old error function

(b) New error function

Figure 4.16. Objective space for different error functions. Using the old error function, G6 is dominated by a group 2 network, and therefore genomes similar to G6 may be difficult to discover. This is not the case with the new error function.



(a) Group 2 genome

(b) Group 1 genome, smaller time window and e-area

(c) Group 1 genome, larger time window and e-area

Figure 4.17. A comparison of the differences between e-area (area beneath the error-time curve) yielded by different genomes and different time windows.

previous, genomes of any group that are not restricted by activation will develop all of their connections immediately so that connection weights converge as early as they possibly can. The group 2 networks found in these preliminary trials behave like this and answer [1, 0] (true) for all questions and yield their lowest possible error very early in development (fig. 4.17a). Using the current error function, the e-area and number of updates for G6 were greater than those produced by a group 2 genome unconstrained by activation, causing the latter to dominate the former (fig. 4.16a). This problem was not encountered in chapter 3 because time window and therefore e-area was low enough for group 2 networks to be dominated by activity-dependent group 1 (zero-error) genomes. However depending on factors such as the time window, observation sequence and the number of questions asked, there is always a possibility that this problem will be encountered in different scenarios. In order to make this problem less likely to occur, a new error function was devised that assigned even greater error to group 2 networks.

The previous scoring function (section 3.3.4) did not necessarily ensure that group 2 networks, which always provide the same answer for any predicate, were negatively weighted. It only penalised networks that always answered [0,0] (unknown) and rewarded correct answers. The new function ensures that networks that always provide the same answer for a predicate receive a lower score than those for which the set of answers have some variety, even if all of those answers are incorrect. The score of each predicate was measured separately so that the total accuracy was set to the total score of all predicates (equation 4.1). a is now a set of matrices, each of which represents a predicate's answer matrix P. This new predicate scoring function enforced an ordering on answer types (equation 4.2). If the answer to a predicate's questions is always [0,0] (unknown), then it is likely that no input connections to that predicate's node cluster have been formed, so a fixed score of 0 is assigned to that predicate (an exception is enforced by the condition '$T_i \neq [0,0]$ for some T_i', explained below). If the network always answers [1,1] (contradiction), this indicates that connections have been formed and that activity is at least reaching the predicate's collectors, so a score of 1 is assigned. If the network always answers [1,0] (true) or [0,1] (false), as with group 2 networks, there is at least some variety in the collector activations and a score of 2 is awarded for that predicate. Finally, when more than one answer is provided for a predicate, the network has demonstrated its ability to distinguish between at least two questions on that predicate and a score of 3 plus the number of correct collector activations (score function S_1 from equation 3.3 on page 94) is assigned. Assigning scores to individual predicates in this way allows for a more gradual increase in fitness as variety is discovered in a greater number of predicates.

$$Accuracy(a) = \sum_{p=0}^{n} S(a_p) \tag{4.1}$$

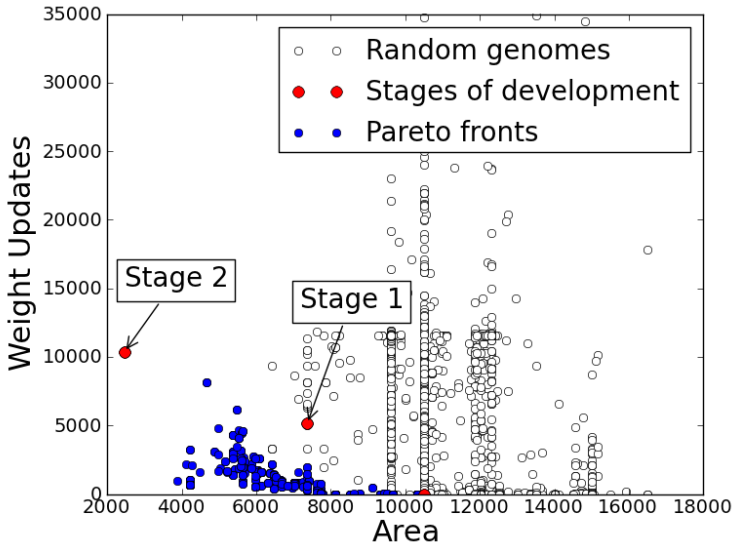
$$S_3(P) = \begin{cases} 0 & \text{if } P_i = [0,0] \text{ for all } P_i \text{ and } T_i \neq [0,0] \text{ for some } T_i \\ 1 & \text{if } P_i = [1,1] \text{ for all } P_i \\ 2 & \text{if } (P_i = [1,0] \text{ for all } P_i) \text{ or } (P_i = [0,1] \text{ for all } P_i) \\ 3 + \sum_{i=0}^{m} S_1(P_i) & \text{otherwise} \end{cases} \tag{4.2}$$

Note in equation 4.2 that the condition '$T_i \neq [0,0]$ for some T_i' enforces an exception when penalising answers of [0,0] (unknown). T is the target matrix for predicate P, and the exception ensures that predicates are not penalised when an answer of [0,0] (unknown) for all of that predicate's questions is in fact the desired output ($T_i = [0,0]$ for all T_i). The questions in A and B question sets (appendix A.1) are selected to test the consequent of every relation as explained in chapter 3.1. However some predicates are not the consequent of any relations and therefore any questions pertaining to them in the A and B sets can only be answered 'unknown'. For example, predicates $P(x,y,z)$, $Q(y)$ and $T(x,y)$ in the NoNeg8 set (fig. 4.8, page 128) are only ever antecedents in any relation and no questions expected to be answered 'true' or 'false' are asked of them in the A and B question sets (fig. A.22, appendix A.1). When all questions for such a predicate are correctly answered as 'unknown', that predicate receives the maximum score ($3 + S_1(P)$) rather than the lowest (0).

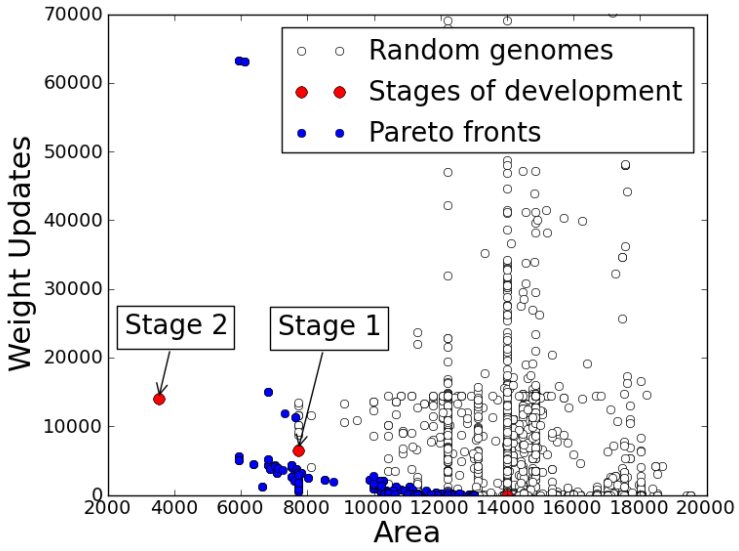
Using this new scoring function, group 2 and 3 networks are always considered weaker than networks that distinguish between questions when providing answers. Therefore, group 2 and 3 networks are weighted less than before in relation to networks that produce different sets of answers, and it becomes easier for a zero-error network that learns slowly to outperform a group 2 network that learns quickly. Using this new scoring function, G5 now yields an e-area of 3522, G6 yields an e-area of 5622 and the group 2 genome yields an e-area of 10320 (fig. 4.16b, page 138). Like the other two points in the plot, G6 is now non-dominated so that it is more likely to be found in the highest rank of the final population.

4.3.5. Results of evolving mediator structures

This section presents the results of attempting to evolve mediator structures using the configuration described in the previous section, including the new error function. The conclusion of these experiments was that although the NSGA-II evolutionary search behaves as expected by gradually improving the Pareto front over time, the discovery of SHRUTI's mediator structures through evolution is highly unlikely because the fitness landscape is heavily populated with group 3 genomes that always answer [0,0] (unknown) and contains insufficient information to identify useful components that when combined in later generations will produce fitter genomes. The necessity for discovering individual components in combination demonstrates the brittle nature of SHRUTI structures. In other words, a SHRUTI structure requires all of its components to be in place in order



(a) NoNeg7



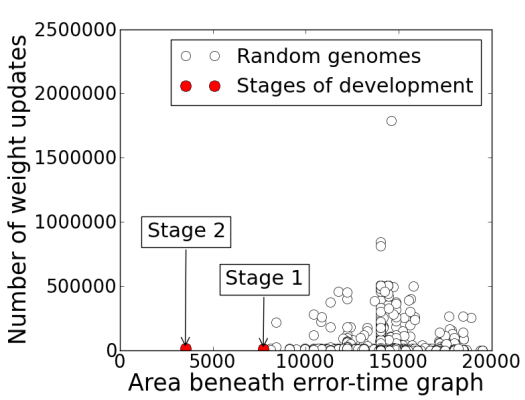
(b) NoNeg8

Figure 4.18. Final Pareto fronts for 20 trials after 500 generations, compared against the fitnesses of 50,000 randomly generated genomes and stages 1 and 2 of the development of genome G5. The random points and developmental stages were generated separately from the evolutionary search and are included in this plot for the sake of comparison.

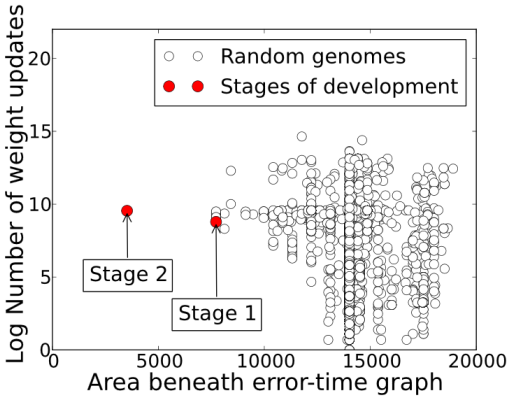
to function properly and therefore does not degrade gracefully. A lack of graceful degradation is a common criticism of localist and symbolic models [77]. Because of the lack of information in the fitness landscape, the evolutionary search could not discover any group 0 genomes[2]. The results that support this are presented and discussed below.

20 evolutionary trials of 500 generations were performed for the NoNeg7 and NoNeg8 logic programs (fig. 4.8). Only 20 trials were performed (as opposed to 50 in chapter 3)

[2]The reader is reminded that as explained in section 4.3.4, the term 'group 0 genome' is introduced to refer to a genome that produces zero-error networks for logic programs that contain conjunctive and non-conjunctive relations, as opposed to group 1 genomes, which only produce zero-error networks for logic programs containing non-conjunctive relations.



(a) Standard scale



(b) Logarithmic scale

Objective pair

	e-area	#Updates	Percentage
1	14015	0	75.34
2	14015	14	4.01
3	14015	40	3.48
4	14015	340	3.04
5	14514	0	1.59
6	12217	14445	1.32
7	14861	14445	1.13
8	18007	0	0.31
9	17508	0	0.23
10	14015	217	0.22

e-area

e-area	Percentage
14015	91.69
12217	1.76
14514	1.71
14861	1.44
17508	0.49
18007	0.36
14312	0.20
17009	0.18
14411	0.17
14465	0.16

Number of Updates

#Updates	Percentage
0	78.45
14	4.13
40	3.52
340	3.06
14445	2.55
26	0.25
217	0.22
3	0.21
123	0.20
12	0.14

(c) 10 most common objectives and objective pairs, ranked by percentage of population occupied.

Figure 4.19. Objective pairs yielded by 50,000 randomly generated genomes. The vast majority are group 3 genomes that yield an e-area of 14015 as a result of answering [0,0] for all questions.

because of the complexity of evaluating larger network structures. Given that each fitness evaluation could take up to 2 minutes (table 4.7, page 137), evaluating 100 genomes every 500 generations had the potential to take as long as 100,000 minutes, or 1667 hours. Figure 4.18 shows points from the final Pareto fronts of each trial, compared against the fitnesses yielded by stages 1 and 2 of development for G5 (red dots) and a set of 50,000 randomly generated genomes (white dots). As explained in section 4.3.4, group 1 genomes are equivalent in behaviour to stage 1 of development, and group 0 genomes are equivalent to stage 2. The experiments found that the evolutionary search was capable of discovering group 1 genomes and therefore performing at least as well as it could in chapter 3, but incapable of discovering group 0 genomes. Although a group 0 genome is never discovered throughout the 20 trials, it is clear that the evolutionary algorithm is behaving as it should do for three reasons: it improves upon the cloud of random points, there is a steady increase in the hypervolume of the Pareto front, and the algorithm is able to discover group 1 genomes. Each of these three points are elaborated on below.

For NoNeg8, figure 4.19 shows the full cloud of randomly generated points on standard and logarithmic scales and table 4.19c shows the most common objective values yielded by the genomes generated. 75.34% of genomes generated are group 3 networks with no connections that answer [0,0] (unknown) for all questions and yield an e-area of 14015. Note that because the new error function (equation 4.2, page 140) makes some exceptions

when penalising answers of [0,0] for some predicates, group 3 networks do not necessarily yield the lowest error and therefore do not necessarily yield the lowest e-area as was the case in chapter 3. Some group 3 networks may contain some number of connections assembled in a meaningless fashion so that none of them lead to the activation of collector nodes and thus output the same as those with no connections. Whether or not group 3 genomes produce connections, they are nonsensical combinations of conditions and actions and make up 91.69% of the randomly generated genomes. Only 0.03% of genomes generated produce group 1 networks, and no group 0 genomes were generated at all. The cloud of points is representative of the space that the evolutionary search is likely to explore. Figure 4.18b shows that the search pushes the boundaries of random cloud, demonstrating its ability to not only search it but also improve on it. However, the search never reaches group 0 genomes that yield the objective values of stage 2 of development for G5, and does not even explore any of the surrounding space. The group 0 genomes are far separated from the cloud and this emphasises how difficult it is for them to be discovered.

Note that the plot of random genomes contains a number of line-shaped artefacts in the form of recurring values for either e-area or the number of weight updates. The behaviour of a SHRUTI network is quite discrete and therefore specific values are likely to reoccur. Any value for e-area will not necessarily always yield the same number of weight updates because as explained for the group 3 networks, connections may develop that do not participate in propagating output. Similarly, some values for number of updates reoccur but are not always associated with the same e-area. Even if two networks produce exactly the same connections and are trained on the same event sequence, the initial weight values for those connections will vary between networks. Therefore the same will be true of connection weight values at any point in time, which nodes are activated at any point in time, and the resulting e-area.

Another interesting artefact is the relative lack of networks with more than a certain number of updates. For example, randomly generated genomes for NoNeg8 produce more networks that yield 14445 updates or less than networks that yield more. Networks with 14445 updates are those in which all nodes that pre-exist before development (predicate and entity nodes) are connected to each other at the first event observation. In other words, among the initial set of nodes, every possible connection is produced and 14445 is the maximum number of weight updates that can be performed on this set. In order to yield any greater number of weight updates, more connections must exist, for which more weight updates can be performed. However, in order to produce more connections, the genome must produce more nodes between which to form those connections. Such genomes are indeed generated at random but due to the nonsensical nature of the randomly generated rules that trigger node production, it is difficult to constrain when node production occurs without constraining it completely. In an unconstrained genome, it is highly likely that a large number of nodes will be produced at the first event observation. When the number of nodes grows to be too high, a penalty is incurred and development stops before connections can be trained. An untrained network will always answer [0,0] and yield the e-area of 14015. In summary, most randomly generated genomes that produce new nodes will yield group 3 networks by incurring a penalty at the beginning of development.

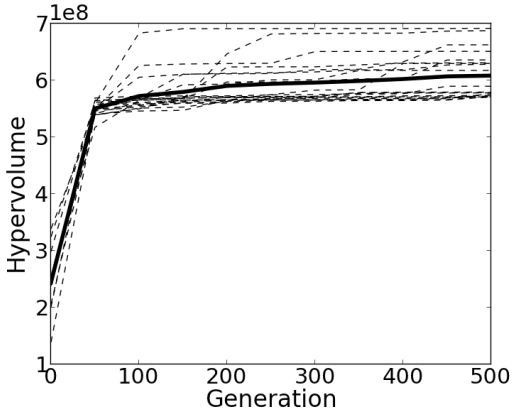
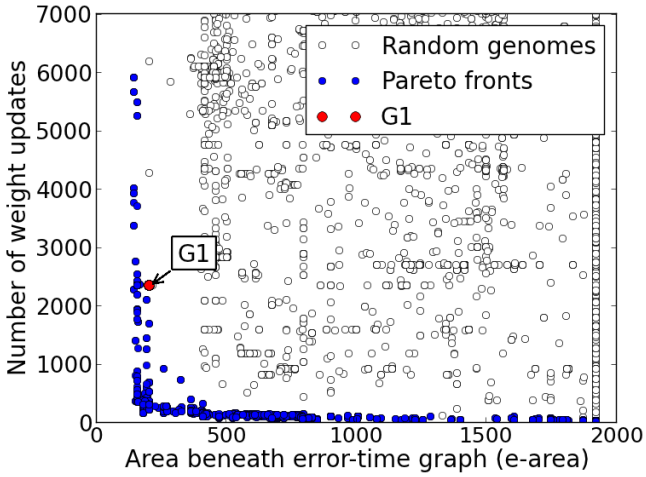


Figure 4.20. The change in hypervolume over 500 generations for all 20 trials for NoNeg8. The thick black line represents the average change in hypervolume, which converges in the earlier generations, thus demonstrating that the Pareto front itself has converged.

The *hypervolume* of a Pareto front is measured as the area of space it dominates and an increase in hypervolume indicates an improvement to the Pareto front in the form of one or more new non-dominated points. Fig. 4.20 shows the hypervolumes measured at 50 generation intervals over all 20 trials for NoNeg8. On average, the hypervolumes continue to increase across all 500 generations, suggesting that the Pareto front is successfully improving over time and therefore that the evolutionary algorithm is behaving as it should do. However, the average front converges even though the Pareto front has not progressed to the neighbourhood of group 0 genomes (stage 2 of development for G5 as shown in fig. 4.18b). On average the fronts are still improving, but considering the average convergence speed and the distance that must be travelled to reach group 0 genomes, it could be an intractably long time before these genomes are discovered.

Nonetheless, the search does discover group 1 genomes, capable of correctly answering questions on non-conjunctive relations. This is shown in fig. 4.18 by the fact that the final Pareto fronts contain points within the region of stage 1 of development, equivalent in behaviour to group 1. The evolutionary search therefore performs at least as well as it did in chapter 3. Fig. 4.21 shows the final Pareto fronts obtained for NoNeg4 from chapter 3 in relation to the target genome and a cloud of 50,000 random genomes. Here also, group 3 genomes, most of which yield an e-area of 1920, occupy a large percentage of the randomly generated population and group 1 genomes occupy only a tiny percentage, suggesting that the former is easy to discover in an evolutionary search and that the latter is more difficult. Nonetheless, the search pushed the boundaries of the random cloud and found group 1 genomes and many improvements to them.

The ease of discovering group 1 genomes when compared to group 0 genomes is owed to the fact that they are much less complex. Both require rules for testing node function equality, which may be possible with one condition that generalises across all node functions as in chapter 3 ($SELF.Function = P\_INPUT.Function$), or may also be possible by considering each node function separately as is necessary for constructing mediator structures in this chapter, which requires five conditions (fig. 4.13, page 132). Function equality can be tested using either approach for group 1 networks, but the second, five-



(a) Final Pareto fronts for 50 trials after 500 generations for the NoNeg4 data set, compared against the fitnesses of 50,000 randomly generated genomes and genome G1.

	Objective pair			**e-area**		**Number of Updates**	
	e-area	#Updates	Percentage	e-area	Percentage	#Updates	Percentage
1	1920	0	62.89	1920	70.64	0	62.89
2	1566	7503	6.11	1566	7.40	7503	12.53
3	1499	7503	2.75	1499	4.06	5913	1.23
4	1487	7503	0.64	1563.5	1.11	7395	0.92
5	460	5913	0.59	418	0.94	12979	0.77
6	418	7503	0.56	460	0.91	13224	0.70
7	1563.5	7395	0.48	1487	0.85	78661	0.53
8	1920	3090	0.42	1172	0.55	14320	0.47
9	1920	12979	0.42	798.5	0.55	11987	0.42
10	1920	13224	0.39	1608.5	0.51	3090	0.42

(b) 10 most common objectives and objective pairs.

Figure 4.21. Randomly generated genomes for the NoNeg4 logic program from chapter 3

condition approach is necessary for group 0 networks as argued in section 4.3.2. Although the search in this chapter discovers group 1 networks, the discovered genomes test function equality with only one condition, but in order to eventually mutate into group 0 genomes, the five-condition alternative must be discovered. However, discovery of the five-condition alternative makes no difference to the objective fitness of group 1 genomes because the behaviour remains the same: connections are formed between nodes of the same function and all questions dependent on non-conjunctive relations are answered correctly. Therefore there is no indication that the evolutionary search has made a necessary discovery when it discovers the five-condition test for function equality. This is what it means for the fitness landscape to be deceptive. The problem of deceptive landscapes is a common one for generative and developmental systems (see section 2.5.3, [122]).

In summary, the increase in hypervolume, improvement on the random cloud and discovery of group 1 genomes all suggest that the NSGA-II algorithm is behaving as it should do. However the results presented above suggest that the target group 0 genomes are so far separated from other points in the fitness landscape that discovery is highly unlikely. In particular, the landscape is heavily populated with group 3 genomes and highly deceptive in that genotypic changes necessary to discover group 0 genomes are not reflected by the

objective values they yield. In general, the fitness landscape lacks the information necessary to influence the discovery of fitter genomes. An informative fitness landscape enables a genetic algorithm to recognise individual innovations that upon recombination may produce fitter genomes in later generations. According to the building block hypothesis [36], genetic algorithms discover fitter genomes at each generation by discovering individual components (or 'building blocks') of above-average fitness and recombining them. This idea is supported by the schema theorem which states that certain building blocks following a similar pattern (or schema) of above-average fitness increase exponentially over time [44]. In summary, the discovery and proliferation of individual components is necessary for the discovery of fitter genomes in later generations, and in order for these components to be discovered their discovery must be reflected by an above-average objective fitness. This was not the case in the evolution of mediator structures, as necessary building blocks for the construction of SHRUTI networks such as the five-condition test for function equality could not be identified. The following section presents the results of a number of experiments that were performed to understand the fitness landscape even further and support these arguments by demonstrating why the discovery of group 0 genomes is so difficult.

4.3.6. Fitness landscape

It is impractical to try and enumerate every possible genome in order to completely map the fitness landscape and demonstrate the points made above. However, it is possible to explore the fitness landscape surrounding the target genome G5 and demonstrate how brittle it is, how easily group 3 networks are found in the fitness landscape, and therefore how difficult it is for the evolutionary search to navigate the fitness landscape effectively so that it has a good chance of discovering G5 or similar genomes. The random sampling in the previous section already demonstrates that the landscape is largely populated by group 3 genomes and that the vast majority of points are far separated from the target G5 genome. Three experiments were performed to understand the fitness landscape surrounding G5 even further. The first inhibits certain rules in the genome in order to isolate others, each of which constructs a different component of the SHRUTI architecture. This experiment demonstrates that certain structures need to be discovered together and not in isolation in order to produce genomes that exhibit the target behaviour. The second experiment performs controlled mutations in which the argument for each condition is mutated into each of its possible values. This demonstrates that only a precise combination of values leads to the construction of the target group 0 network, otherwise the genome will mostly produce group 3 or sometimes group 2 networks. The third experiment produces many random mutations of G5 and attempts to mutate the offspring back into G5. This experiment essentially demonstrates the same point as the second, but also shows how unlikely it is to discover G5 genomes from random mutations of its neighbours and therefore that even close genotypic proximity does not imply close proximity in terms of objective fitness. It is this lack of information that makes the fitness landscape deceptive and therefore difficult to navigate. All following experiments were performed using NoNeg8 (fig. 4.8, page 128) as the logic program to be represented by developed networks.

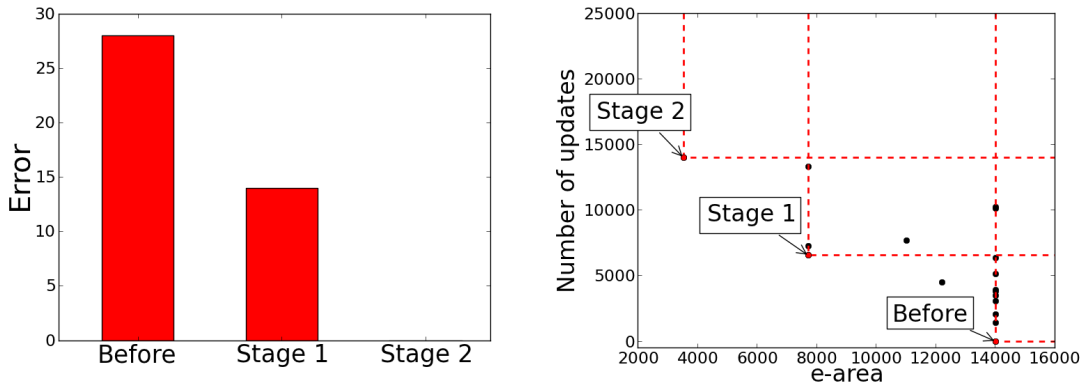
Inhibitory experiments

Experiments were performed by inhibiting certain rules so that individual rules or sets of rules could be observed in isolation. Table 4.8 presents objective values produced by the networks developed when inhibiting different sets of rules in the genome and fig. 4.22b presents a plot of these values.

As explained in section 4.3.2, genomes G5 and G6 develop networks in two stages. Before development, all questions will be answered 'unknown'. In the first stage, non-conjunctive structures are formed and questions pertaining to these can be answered correctly. In the second stage, conjunctive relational structures are formed and all questions can be answered correctly. At each stage, there is a reduction in e-area and an increase in the number of updates, both in the direction of their final target values as shown in fig. 4.22. Although the goal is to *minimise* the number of updates, at each stage the e-area is still reduced and together both objective values move in the direction of G5. This suggests that the path along which the two steps are discovered follows that of the objective fitness. In generative and developmental systems (GDS), rediscovery of particular genomes is difficult because this is not always the case; movement in the correct direction in genotype space does not necessarily imply movement in the correct direction in objective space [122].

Although the discovery of each stage of development reflects progress in objective fitness, individual discovery of the rules necessary to implement each stage does not, as is true of GDS. Necessary steps in rule discovery that the evolutionary search must take in order to pass between stages are represented by the unlabelled points in fig. 4.22b, and table 4.8 shows exactly which combinations of SHRUTI's structures yield these objective values.

From the second and third sub-tables, it can be seen that no discovery of the mediator structure for any single node function alone leads to a reduction in error or e-area when compared to a network with no connections. In fact, the number of weight updates is increased and so performance is weaker in terms of Pareto dominance. In both cases, rules constructing mediators for collectors and enablers must be discovered together in



(a) Error (b) Objective values

Figure 4.22. The change in objective values as each stage of mediator development is discovered. Plotted values are taken from table 4.8. Black dots represent points from the table that do not represent complete stages.

Before development

Component	Error	e-area	#Updates
None	28	14015	0

Non-conjunctive structures only

Mediator structure	Error	e-area	#Updates
Collectors	28	14015	3081
Enablers	28	14015	1413
Role nodes	28	14015	2050
Collectors & enablers	24	12217	4494
Collectors & role nodes	28	14015	5131
Enablers & role nodes	28	14015	3463
All (stage 1)	14	7222	6544

Conjunctive and non-conjunctive structures

Mediator structure	Error	e-area	#Updates
Collectors	28	14015	3782
Enablers	28	14015	3885
Role nodes	28	14015	6340
Collectors & enablers	20	11017	7667
Collectors & role nodes	28	14015	10122
Enablers & role nodes	28	14015	10225
All (stage 2)	0	3522	14007

Transition between stage 1 and stage 2

Mediator structure	Error	e-area	#Updates
Stage 1 (no conjunction)	14	7222	6544
Conjunction for collectors	14	7722	7245
Conjunction for enablers and roles	14	7722	13306
Stage 2 (target genome)	0	3522	14007

Table 4.8. Objective fitness values obtained when discovering different combinations of rules for constructing mediator structures for the NoNeg8 logic program, for which the maximum error is 36. However, the error yielded by a group 3 network is 28.

order for any improvement in error or e-area to take place, as such genomes produce the smallest combination of connections between predicate clusters that can yield any change in fitness: group 2 networks that always answer 'true' for any particular predicate. Even if group 1 networks are discovered before group 2 networks, the same is true because group 1 networks are an extension of group 2 networks (page 108), and so collector and enabler mediators must still be discovered together. The discovery of stage 1 of development cannot take place until this has happened.

The fourth sub-table shows the difficulty in making the necessary discoveries to progress from stage 1 to stage 2. Enablers and role nodes share a regulatory sub-network for the development of mediator structures (fig. 4.13) and therefore disabling the rule that produces conjunctive structures for one disables it for both. Evolution must discover medTr mediators for collectors and medQu for enablers and role nodes. Discovering either conjunctive structure in isolation does not improve error or e-area from stage 1 and increases the number of updates with the result that stage 1 genomes have Pareto dominance over these discoveries. Once again, both substructures must be discovered together for any reduction to be made in error and e-area.

In summary, rules for constructing certain components of a relational structure must be

discovered together in order for any progress to be made towards the Pareto front. If they are discovered individually, the resulting networks are Pareto-dominated by their parents by virtue of a greater number of weight updates and no improvement to e-area. Even if the number of updates were to be removed as an objective so that the search was only minimising e-area, there would still be no improvement to e-area when necessary rules are discovered individually and therefore no information to indicate a correct change in the direction of the target fitness. The probability of discovering two components together is considerably less than that of discovering them individually because the former is the product of the individual probabilities. This provides part of the explanation as to why mediator structures are so difficult to discover in the evolutionary search.

Table 4.8 contains 17 unique combinations of rules, 11 of which yield an error of 28, that of group 3 genomes that always answer [0,0] (unknown). Already this begins to show how easily a group 0 (zero-error) genome can be mutated into a group 3 genome. The following experiments support this further.

Controlled mutation

The next means of exploring the landscape surrounding G5 and other group 0 genomes was to mutate the arguments for each condition in the genome separately in a controlled fashion, exploring each of their possible values. Figs 4.23 and 4.24 show the results and demonstrate that for most conditions, only one particular value yields an error of zero and the corresponding e-area. Therefore only a precise combination of argument values for conditions in a genome will produce zero-error networks.

Fig. 4.23 shows what happens when conditions for node function are mutated so that they test for different functions. For SHRUTI to function properly, it is very important which nodes connect to which. Disrupting connections between nodes of just one function is sufficient to destroy the functionality of the network as a whole so that no questions can be answered. In most cases, error increases to 28, that of group 3 networks that always answer [0,0] (unknown). Again, this demonstrates how easily group 0 genomes are mutated into group 3. In other cases, error increases to 20, that of group 2 networks that always answer 'true' or 'false' for each predicate. In both cases, e-area increases and the number of updates drops. When mutating a condition to test for mediator nodes (medTr or MedQu), new mediator nodes are recursively created for each existing mediator node if the condition evaluates 'true', and in most cases the size of the network grows beyond its maximum size so that the genome receives a penalty, shown by white squares in fig. 4.23.

Fig. 4.24 (page 151) demonstrates the effects of mutating conditions on the number of indirect inputs or outputs, the conditions which affect whether conjunctive or non-conjunctive structures are produced. Condition 5 affects the construction of collector mediators medTr and condition 7 affects the construction of enabler and role node mediators medQu. Mutating the former has a negative effect on error and e-area, but mutating the latter does not. $nIndirect\_I/O = x$ will always evaluate false for any $x > 0$ at the start of development, and when it does it expresses the action for constructing conjunctive mediator

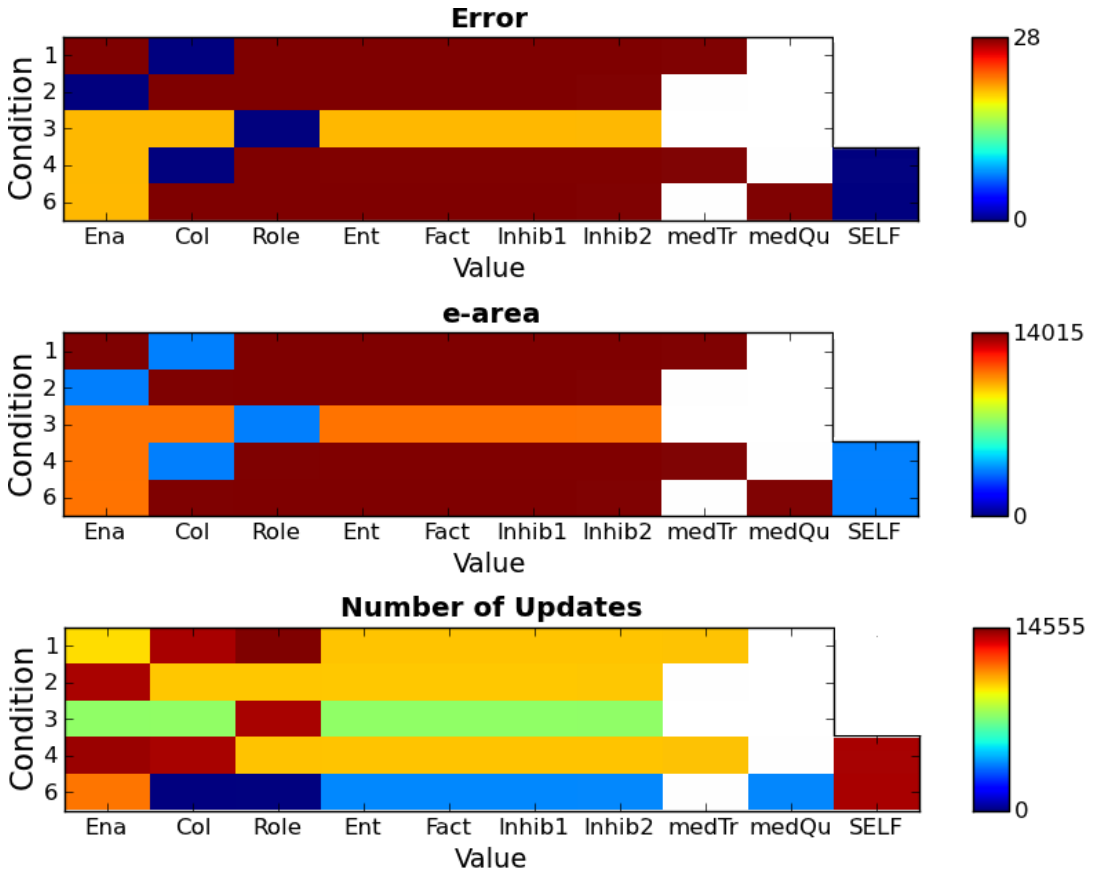


Figure 4.23. Controlled mutation of conditions on node function. The y-axis represents conditions from G5 (fig. 4.13) that pertain to node function, and the x-axis represents values that the arguments of those conditions may take ($SELF.Function = Ena$, $SELF.Function = Col$, etc.). The colour of each square represents the objective value yielded by the network that develops when the argument of the condition in the y-axis is set to the value in the x-axis. White squares indicate a penalty incurred when the number of nodes grows too large. Conditions 1-3 are conditions on SELF (e.g. $SELF.Function = X$) and conditions 4 and 6 are conditions on P_CON (e.g. $P\_CON.Function = X$). Conditions on P_CON may take SELF as an argument ($P\_CON.Function = SELF.Function$), but conditions on SELF may not, hence why no results for SELF as an argument are provided for conditions 1-3.

nodes. Therefore when $x > 0$, only conjunctive mediator structures will be produced, and no non-conjunctive mediators. Conjunctive medQu nodes are still able to propagate activations of consequent enablers and role nodes to their antecedent counterparts without the non-conjunctive mediators. For example, if $A \rightarrow C$ is true, $A \wedge B \rightarrow C$ is also true. Even if $A \rightarrow C$ is to be queried but the structure for that relation does not exist, a $A \wedge B \rightarrow C$ structure does because G5 produces every possible conjunctive mediator. The medQu nodes for this relation can propagate activation of enablers and role nodes from C to A in place of the missing medQu nodes for $A \rightarrow C$ because medQu nodes can be activated with a threshold of one input. Therefore querying C still queries A. The inference process is not disrupted by missing non-conjunctive medQu nodes and so zero-error can still be achieved. However this is not the case when mutating condition 5 because conjunctive medTr nodes do require both inputs to be active and so cannot act as surrogates for non-conjunctive medTr nodes. Mutating condition 5 does disrupt the required propagation of activation and consequently some questions will be answered incorrectly.

In all cases, mutating a condition mutates the rule in which it participates. One condition

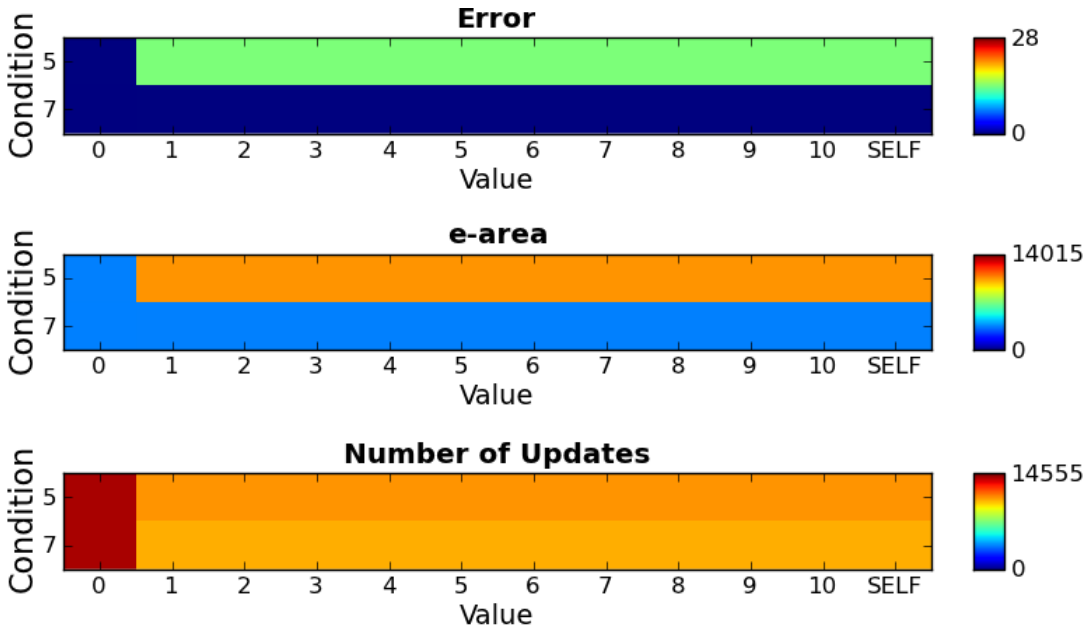


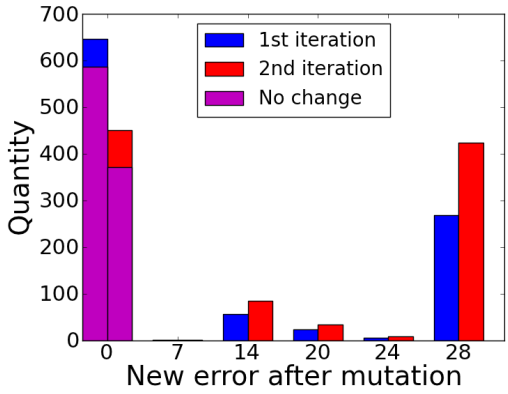
Figure 4.24. Controlled mutation of conditions on the number of indirect inputs or outputs in G5. The y-axis represents conditions from G5 that pertain to the number of indirect inputs or outputs, and the x-axis represents values that the arguments of those conditions may take ($SELF.nIndirect\_I = 0$, $SELF.nIndirect\_I = 1$, etc.).

evaluating as false negates the rule as a whole and has the effect of inhibiting that rule, the effects of which were demonstrated in the previous set of experiments. Not only must the set of rules be in precise combination, but the set of conditions and arguments that define those rules must be also. The results of these experiments also support the argument that the fitness landscape surrounding target genomes is largely populated with group 3 genomes and some group 2 genomes.

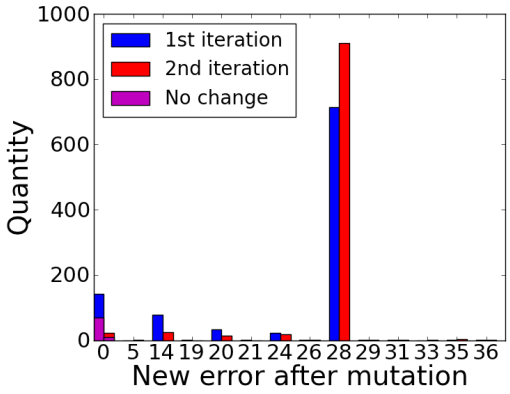
Random mutation

These next experiments also demonstrate the proximity of group 3 genomes to group 0 genomes in genotype space by mutating conditions. However this time the mutations are random, and in addition these experiments attempt and fail to mutate the children of these mutations back into G5 or similar genomes, in order to demonstrate that a landscape of group 3 genomes contains insufficient information to inform the evolutionary search of any genotypic proximity to group 0 genomes.

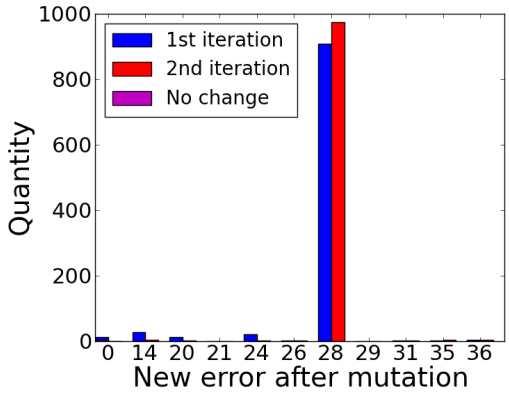
Three trials were performed with three different mutation rates: 0.01, 0.05 and 0.1. For genomes of 25 conditions or actions, these rates were estimated to mutate approximately 0.41, 2.06 and 4.13 exons per genome respectively. For each trial, an initial population of 1000 copies of G5 were produced and subjected to two iterations of mutation. The first iteration was informative of the fitness neighbourhood surrounding G5, and the second iteration was informative of how likely it was that genomes in this neighbourhood could be mutated back into G5 or similar zero-error genomes. The same fixed event sequence was used in each trial. Fig. 4.25 shows the errors yielded by the networks for each mutation rate and table 4.9 shows how many genomes were improved by the second iteration.



(a) pMut = 0.01



(b) pMut = 0.05



(c) pMut = 0.1

Figure 4.25. Error values produced by random mutation of 1000 copies of the genome G5 according to different mutation rates. Two iterations of mutation were performed on each copy of the genome. The majority of mutations result in group 3 genomes that always answer [0,0] (unknown) and yield an error of 28. The maximum possible error is 36

	Number of error reductions	
Mutation rate	**Any**	**To zero-error**
0.01	3	1
0.05	23	1
0.1	13	0

Table 4.9. The number of reductions in error made by each mutation rate in the second iteration of mutation. 1000 genomes were mutated for each mutation rate.

In each case, mutation into group 3 networks that yield an error of 28 in the first iteration was very common. The only exception was for the mutation rate of 0.01, but only because the mutation rate was so low that a large number of genomes were not mutated at all. Only a small number of group 2 genomes (error = 20) were generated for each mutation rate. In the second iteration, even more group 3 genomes were generated. Only a small number of networks have their error reduced by the second iteration and only one or two of these are reduced to zero-error, demonstrating how difficult it is to generate zero-error networks, even from genomes that are close in genotypic space. The results support the hypothesis that the fitness landscape is largely populated with group 3 genomes and is deceptive in that it contains insufficient information to inform the evolutionary search of proximity to genomes of greater fitness.

In summary, the rules and conditions required for group 0, zero-error networks to be constructed must be discovered together and not in isolation, and must behave in a very precise manner. Even points in the fitness landscape that surround these genomes in genotype space do not point towards a target network's fitness because it is heavily populated by genomes that produce group 3 networks. This is true of the fitness landscape in general and even a small change in the correct direction in genotypic space, such as correctly identifying function equality for one node function, cannot improve objective fitness on its own. A more informative fitness landscape is required to identify each of the necessary components or 'building blocks' of complete SHRUTI networks.

These findings explain the lack of success in attempting to discover group 3 genomes. Convergence of the hypervolume indicates that the Pareto front itself has converged (fig. 4.20). It is likely that the Pareto front is difficult to improve further because as the experiments have shown, a large portion of mutations will produce group 3 genomes that produce networks that always answer [0,0] (unknown). These networks occupy a considerably large portion of the objective fitness landscape as was also shown by these experiments and by the random sampling in the previous section (fig. 4.19). The target genome lies at a point in the fitness landscape distant from most others generated by the random sampling (fig. 4.18b). Necessary changes in genotypic space to move to this remote point in the fitness landscape are not reflected by the objective fitness values they yield because components of the target genome must be discovered together, and not individually, in order to influence objective fitness. Even if a genome is close to a group 0 genome in genotype space, it may still be distant in the fitness landscape.

It is reasonable to assume that evolutionary algorithms other than NSGA-II would experience the same difficulties because of the lack of information necessary to identify individual building blocks that improve fitness when combined. In chapter 3, discovery of zero-error networks was much simpler because the fundamental requirements of such genomes was one condition for type equality and one condition for connection addition. This is not the case here, because the genome requires five conditions for function equality and four actions for mediator construction, all in precise combination. The SHRUTI genome provides an explicit description of the SHRUTI structure by providing explicit descriptions of each of its components, and so it is reasonable to assume that the problems encountered with evolving SHRUTI networks are not caused by the genome representation but by the structure of the SHRUTI networks themselves. No matter how the SHRUTI networks are represented in a genome, they require interconnections between nodes of each function that need to be discovered in unison, and they cannot yield the functionality of zero-error networks without them. This demonstrates that SHRUTI structures do not degrade gracefully. This is a common criticism of localist and symbolic systems [77].

In conclusion, although genotypic representations of SHRUTI's mediator and conjunctive structures exist, such genomes are not easily discovered through evolution due to the brittle nature of SHRUTI networks. The following section demonstrates that the same is true of SHRUTI's fact structures.

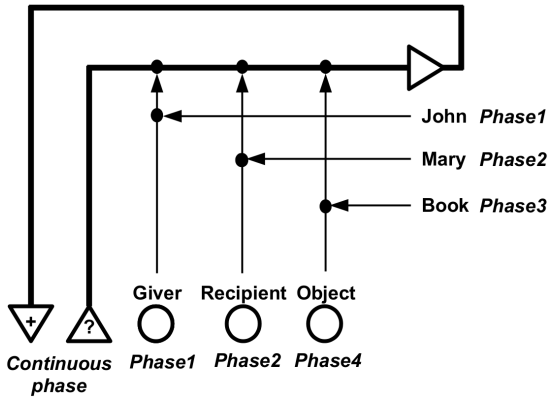
4.4. Fact structures

This section explores the representation of fact structures in developmental genomes, and the difficulties in evolving these genomes. A diagram of SHRUTI fact structures may be found in fig. 2.8 on page 44. Although improvements were made to SHRUTI's fact structures in the form of the SMRITI model [91, 94], the work in this section focuses on the development and evolution of the original SHRUTI fact structures for simplicity and because they have been used in all other experiments conducted thus far. Nonetheless, results from developing and evolving the simpler SHRUTI structures were indicative of potential difficulties in attempting to do the same with larger, more complicated SMRITI structures due to the similarity between the two models. SMRITI involves the same representational requirements as SHRUTI because both act as temporal pattern-matching circuits for predicate clusters. The main difference is that the circuits used for matching bindings in SMRITI are represented by individual nodes in SHRUTI. SMRITI contains circuits for receiving entity-role instantiations from role and entity nodes, detecting binding errors (inhib1 nodes in SHRUTI), confirming binding matchings (inhib2 nodes in SHRUTI) and propagating the assertion to predicate clusters (fact nodes in SHRUTI).

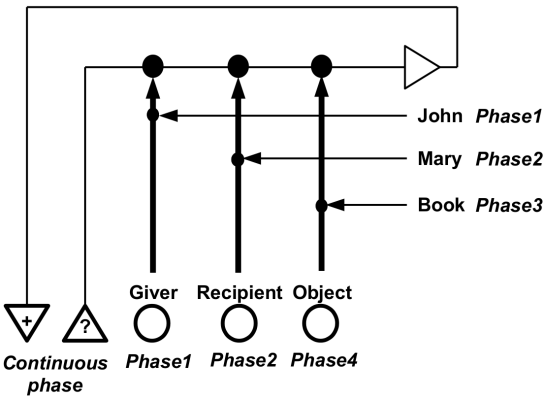
However, no learning algorithm was ever proposed for SHRUTI facts, only for SMRITI [91]. Before a genome for SHRUTI facts could be designed, a learning algorithm for the simplified fact structures that behaves similarly to the learning algorithm employed by SMRITI was produced and is described in section 4.4.1. The genome model for fact structures is proposed in section 4.4.2 and tested in section 4.4.3. Extending the genome model to represent facts in this way further supports the claim of SHRUTI's developers that SHRUTI in general can be represented using indirect encoding [96], which gives it another aspect of biological plausibility. Evolution of fact genomes is attempted in section 4.4.4, but once again genomes that produce zero-error networks were not found. To demonstrate why, the fitness landscape is explored in section 4.4.5. Difficulties in evolving SHRUTI's fact structures were discovered to be for the same reasons as with mediator structures; a fact structure requires every one of its components to be in place in order to function properly and the fitness landscape contains insufficient information to identify those components individually. Such difficulties evolving the simplified structures suggest that the same would be true of the more complex structures employed by SMRITI, which also use fully localist representation.

4.4.1. Learning fact structures

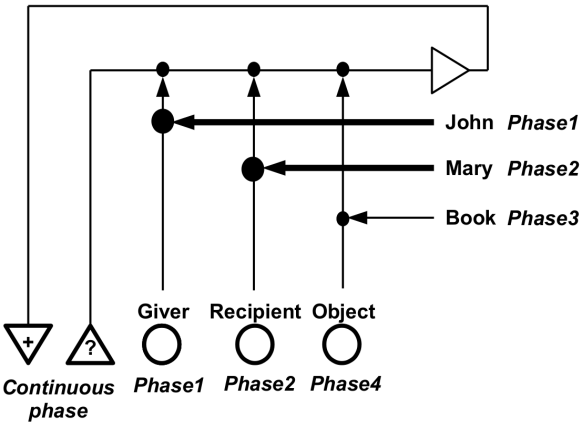
In the original SHRUTI literature, SHRUTI fact structures were claimed to not be amenable to learning and therefore a learning algorithm was not proposed. Only for SMRITI was a learning model proposed, using Long-Term Potentiation (LTP) to recruit neurons in the various binding circuits [91]. Neurons in these circuits are recruited when inputs from role and entity nodes co-activate. Because the nodes in SHRUTI fact circuits are representative of the more complex SMRITI circuits, a learning model using the same idea to recruit SHRUTI fact neurons is introduced in this subsection. Although the learning model is not



(a) Learning the enabler-fact-collector path when the enabler (?) and collector (+) nodes coactivate.



(b) Learning inhibitory inputs from role nodes to inhib2 nodes when enabler (?) and role nodes co-activate.



(c) Learning inhibitory inputs to inhib1 nodes when entity and role nodes fire in the same phase. *Book* and *Object* fire in different phases and therefore the corresponding inhibitory node is not strengthened.

Figure 4.26. Learning fact structures when a predicate instance is observed. Figs. a-c shows each type of connection being strengthened separately. Lines and dots in bold show the connections being strengthened in each figure.

perfect, it is sufficient to allow facts to be learned in conjunction with the development of fact structures in the following subsection. Furthermore, the learning mechanism has no bearing on the evolvability of fact structures because it is not the learning mechanism that is evolved, but the architecture in which the existing learning model takes place.

Facts for each logic program are learned using fixed sequences which repeat each fact three times in the order they are presented in figure 4.8 on page 128. When each fact is observed in fact learning mode, all predicate nodes pertaining to that fact are activated and learning through LTP begins. All connections are strengthened simultaneously, but how each type of connection gains strength will be discussed separately. In the following description, when referring to inhibitory connections, the term *main input* is used to refer to the signal being inhibited and the term *inhibitory input* is used to refer to the signal providing the inhibiting signal. For example, with respect to the first array of inhibitory nodes (inhib1 nodes) in figure 4.26, the *main input* is that provided by a role node, and the *inhibitory input* is that provided by an entity node.

Connections along the path from the enabler to the collector via the fact node are learned as soon as the enabler and collector are activated (fig. 4.26a), since the fact and collector nodes both have a recruitment threshold of one input. Note that the main input to each inhibitory connection has no activation threshold and always propagates activation as long as activation at the inhibitory input does not prevent it. Although the main inputs to inhibitory connections do not require strengthening in order to affect the output of the fact circuit, the inhibitory inputs do and are strengthened when they coactivate with the main input. The main and inhibitory inputs to an inhibitory node are considered to coactivate when they either both fire in the same phase or they both fire and at least one is continuous (that is, not spiking). An inhibitory input in the second array of inhibitory nodes (inhib2 nodes) receives activation from a role node and is strengthened when it coactivates with the main input provided by the output of the enabler (fig. 4.26b). An inhibitory input in the first array (inhib1 nodes) that receives activation from an entity node strengthens when it fires in phase with the main input provided by a role node (fig. 4.26c).

This model includes no means of unlearning facts. This was not covered in the original literature. The SMRITI literature suggests that forgetting could take place in a 'memory consolidation' stage in which important facts are strengthened and unimportant facts are weakened, but an implementation for this was not proposed [92]. Development of such a mechanism was considered for this thesis, but was unnecessary as it would have no bearing on the evolvability of fact structures. Furthermore, it would have been expensive. Unlike the unlearning of relations which can happen in parallel to development, it is proposed that SMRITI's memory consolidation process would occur in a separate stage after the learning and development period akin to the theory that biological memory consolidation occurs during sleep [92]. To test this properly would have required multiple periods of learning/development each followed by the proposed memory consolidation stage, resulting in a much longer training process. With no means of forgetting unwanted facts, fact circuits could only be trained on fixed, noiseless observation sequences and it was assumed that learned facts were always true.

The time window for synchrony was set to 7 observations when learning facts in further experiments, as this allows enough time for activation to propagate from the enabler to the collector.

4.4.2. Developing fact structures

Now that a basic fact learning algorithm had been proposed, the next goal was to produce a genome capable of producing structures that could learn facts. Producing such a genome demonstrated that the representation of SHRUTI structures through the biologically plausible means of indirect encoding extends beyond relational structures and further improves the biological plausibility of SHRUTI networks in general.

Fig. 4.27 presents the genome used to develop fact structures. Unlike the work on genomes for mediator structures, a genome that does not restrict development by activation is excluded from these experiments for reasons which will be explained below. First it is

important to understand how the existing genome works.

Fact structures are composed of three main components: the path from the enabler to collector, the inhibitory connections from role nodes and the inhibitory connections from entity nodes. Each of these components forms an axis of development for the next in a developmental process shown in figure 4.28 on page 158.

Axis 1 - Creating fact nodes: A fact node between the enabler and the collector is formed by rule 1 (fig. 4.28b) and a set of inhib2 inhibitory nodes is formed between the enabler and the new fact node by rule 2 (fig. 4.28c). Each of these new nodes receives 0.9 of its parent's chemicals so that the amount of chemical 2 in each inhibitory node matches that of one of the role nodes representing predicate arguments (see the discussion on choices for chemical 2 values for predicate clusters on page 123).

Axis 2 - Connecting predicate arguments: The connection from the enabler to the fact node forms a new axis for development along which inhibitory inputs to inhib2 nodes may be formed. These new connections are formed from active role nodes by rule 3 (fig.

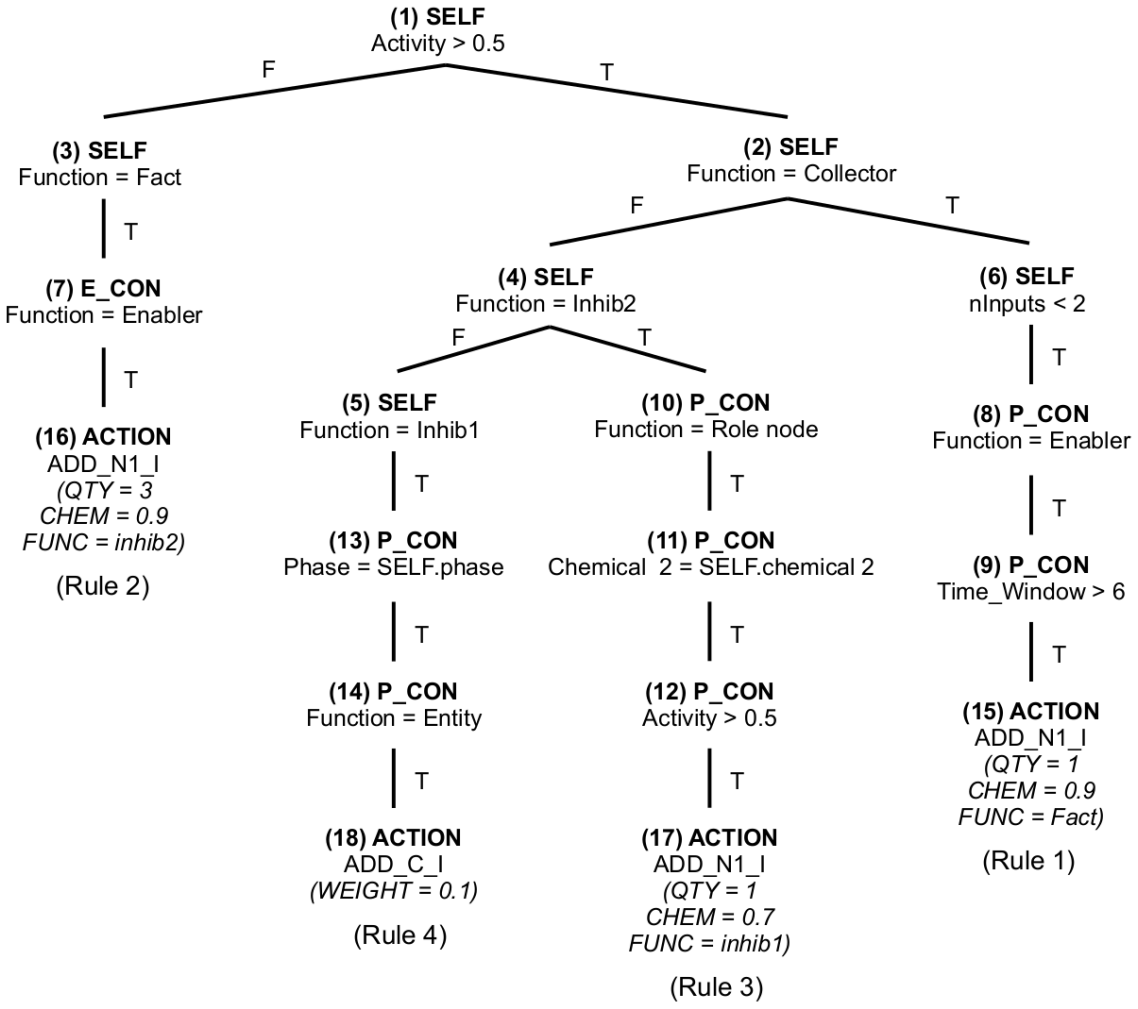
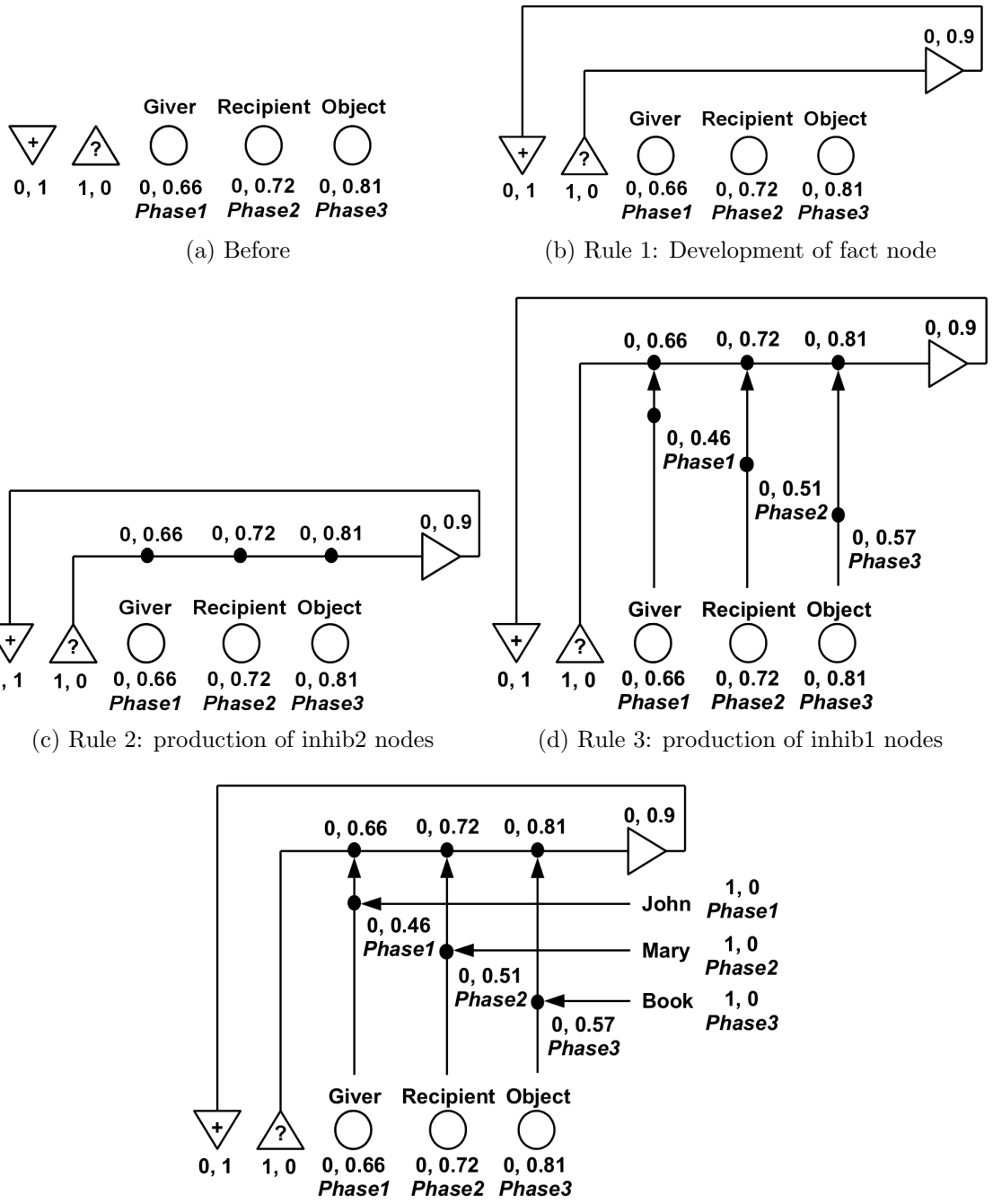


Figure 4.27. Genome for developing fact structures. Terminal nodes correspond to actions, all other nodes correspond to conditions that must be met in order for those actions to be executed, and each path from the root to a terminal node corresponds to a different rule. Each rule creates a different component of a fact structure in the order by which the rules are numbered (fig. 4.28).



(a) Before

(b) Rule 1: Development of fact node

(c) Rule 2: production of inhib2 nodes

(d) Rule 3: production of inhib1 nodes

(e) Rule 4: Connecting inhib1 nodes to entity nodes

Figure 4.28. The development of fact structures as each rule from fig. 4.27 is applied. Numbers displayed by each node correspond to that node's chemical levels (chemical 1, chemical 2).

4.28d) and contain inhib1 nodes that are used to confirm binding matches. Constraining connection formation to nodes of equal chemical levels (condition 11 from fig. 4.27) uses positional information from role and inhib2 nodes to ensure that a separate inhibitory connection is formed for each predicate argument.

Axis 3 - Connecting entity nodes: The new inhibitory connections from role nodes form axes for the development of inhibitory inputs to inhib1 nodes from the entity nodes (4.28e). Rule 4 produces these connections, on the condition that the inhibitory nodes and the entity nodes are firing in the same phase (condition 13).

In summary, a fact structure is created every time a new fact is encountered and the learning algorithm strengthens it further. However, the disadvantage of this approach is that as with the activity constrained genomes for relational structures, a genome behaving in this way is effectively performing the role of the learning algorithm anyway. This was difficult to avoid because of the way facts are represented in SHRUTI.

The complexity of a fact genome unconstrained by activation would be greater than that of a mediator genome unconstrained by activation, so much so that evaluating a network would be intractable. A genome unconstrained by activation would create a fact structure for every possible fact. However because there is no global controller for the developmental process, the genome for a predicate's collector (from where the development of a fact structure begins) has no means of knowing whether or not the fact structure it is about to develop already exists. Because this information is not available to inhibit development, a structure for every possible fact would be created at every time step. The resulting network would contain an extremely large number of nodes, connections and structures, and would be very expensive to evaluate. This problem was overcome for mediator structures because the ADD_N2 actions used by the genome to produce mediator nodes can detect when an intermediary node between two nodes from different predicates exists. However, this only works for one intermediary node and not entire circuits. In order to detect if a certain fact structure already exists, the collector node that triggers its development would somehow have to acquire information from all entity and role nodes of all the fact structures associated with it.

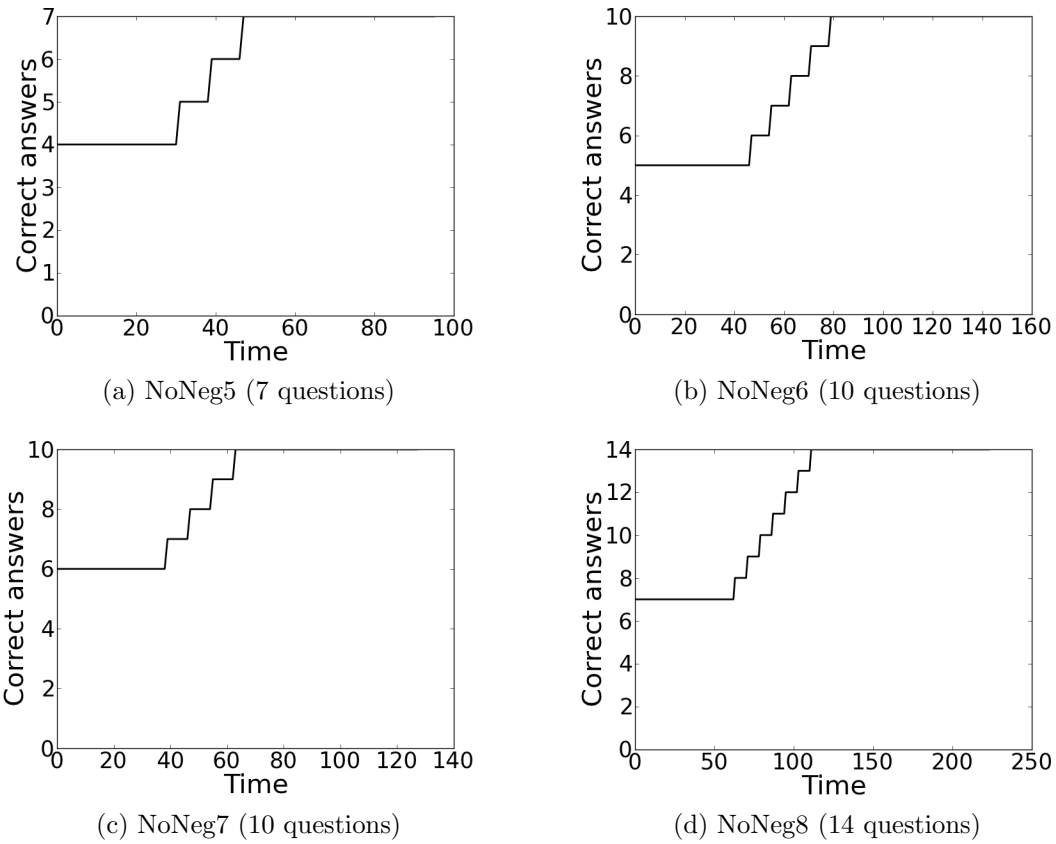
The development of a fact structure must not only be constrained by activation, but it must also be constrained so that structures are only constructed once per activation. Condition 9 in G7 (fig. 4.27, page 157) enforces this by ensuring that construction of the first axis only occurs when the remaining time in the window of synchrony (*Time_Window*) is at its maximum value (7). Inhibitory nodes in this model can only take one inhibitory input, and so this alone ensures that the second and third axes are only developed once for each inhibitory node in the first axis. However care must be taken to ensure that the correct inhibitory connection is formed in each case. Restriction by activation and chemical level equality (conditions 11 and 12) ensure that the correct connections are made for the second array of inhibitory nodes (inhib2), and restriction by phase equality (condition 13) ensures this for the first array of inhibitory nodes (inhib1). Even with these constraints in place, the genome will continue to reproduce a structure for each fact each time it is observed unless there is some constraint on the total number of fact structures that a predicate cluster can be associated with. Condition 6 provides such a constraint by restricting the number of inputs to a collector, the source of fact development.

In summary, activation constraints must be enforced in order to prevent the network becoming so large as to be unmanageable by producing every possible fact structure at every point in time. Although evidence supports activity-dependent development of biological neurons [33], in this case biological plausibility is lost because such constraints result in a genome that essentially copies the learning algorithm thus making it redundant. The same was argued for conjunctive and non-conjunctive relational structures. Once again,

the solution may be to introduce a more distributed representation in which development would behave closer to the information processing theory by improving performance factors [69, 100] instead of producing a specific structure every time evidence is observed. Regardless of what triggers development, a description of a SHRUTI fact structure that can be expressed multiple times is still contained within the genome. In this respect, the production of an indirect encoding for SHRUTI fact structures has been successful and the indirect encoding of SHRUTI in general has been extended beyond relational structures. The following subsection demonstrates that the structures developed by these genomes are capable of learning facts correctly and answering questions about them.

4.4.3. Testing development

The genome was tested by developing the fact structures for NoNeg5-8 data sets (Fig. 4.8, page 128). The genome was trained on fixed sequences of observations in which each fact was presented three times. A maximum time window of 7 was used, and developed fact structures were tested on C questions sets found in appendix A. The test questions contained each fact and one question expected to be answered 'unknown' for each predicate. All test questions were answered correctly for each logic program, as shown in fig. 4.29. This demonstrates that all facts were developed and learned correctly. Table 4.10 shows the statistics for each developed network and demonstrates that like the genomes



(a) NoNeg5 (7 questions)

(b) NoNeg6 (10 questions)

(c) NoNeg7 (10 questions)

(d) NoNeg8 (14 questions)

Figure 4.29. Developing SHRUTI fact structures using genome G7. The graphs show the number of correct answers to questions over time for the NoNeg5-8 data sets. A fixed event sequence was used in each case. Each network developed so that it could answer all of its test questions correctly.

Program	#Facts	#Predicates	#Nodes	#Connections	#Updates
NoNeg5	3	3	72	90	204
NoNeg6	5	4	97	120	288
NoNeg7	4	4	79	84	207
NoNeg8	7	6	99	108	299

Table 4.10. Statistics for each genome developing fact structures for each logic program using fixed event sequences. All test questions are answered correctly in each case. Each logic program and developed network is a different size, despite the fact that the same fixed-size genome is used in each case. This shows that the genome is scalable.

for developing relational structures, the fact genome is also scalable because it develops networks for logic programs of different sizes.

For each logic program, the increase in fitness (the number of questions answered correctly) in fig. 4.29 resembles a series of steps, one for each fact. Following the first presentation of each fact, weights weren't strong enough to answer questions correctly. However, after the second presentation of each fact, the set of weights corresponding to each fact were strengthened so that a new question was answered correctly at each presentation, which explains the step-like nature of the increase in fitness.

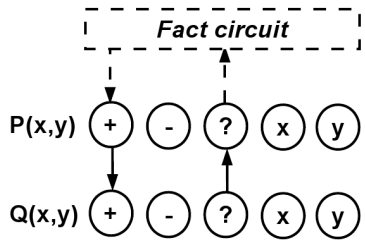
4.4.4. Evolving fact structures

This section presents the results of attempting to reproduce the fact genomes through evolution. Fact genomes were not discovered by the evolutionary search for the same reasons that mediator genomes were not discovered in section 4.3; the target genome required a very precise combination of rules, conditions and actions in order to work. The individual discovery of these components was not reflected by the changes they made to objective fitness, if any.

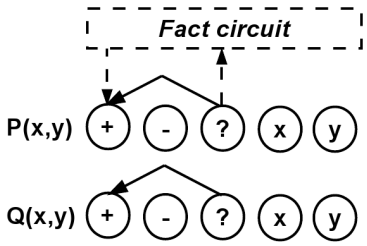
Most of the same parameters used in section 4.3.4 were used in these trials. However genomes were fixed at a size of 30 conditions and actions because the target genome was larger than the genome used for mediator structures. Networks were trained on the full fact sequences since each fact only needed to be presented three times to train the networks and the total training time was short compared to that of mediator structures. Constraints were once again enforced on the developed networks for each logic program. These constraints are listed in table 4.11 and were more than sufficient for developing networks for each logic program using genome G7. Performing many trials of evolution over 500 generations was expensive due to the complexity of evaluating potentially large

Logic program	Time limit (seconds)	Maximum size (nodes)
NoNeg5	30	500
NoNeg6	45	750
NoNeg7	45	750
NoNeg8	60	1000

Table 4.11. Constraints on network size and development time when evolving networks for representing fact structures. Networks which develop beyond these limits are penalised to the lowest rank. As with mediator evolution, time limits were chosen according to the specifications of the machine on which experiments were performed (page 137).



(a) Relational structure evolution: connections formed between predicates.



(b) Fact structure evolution: connections formed within predicates.

Figure 4.30. The difference between group 2 networks for relational structures and fact structures. Both configurations result in the same answer (in this example, 'true') always being provided for a particular predicate (see table 3.11 on page 105).

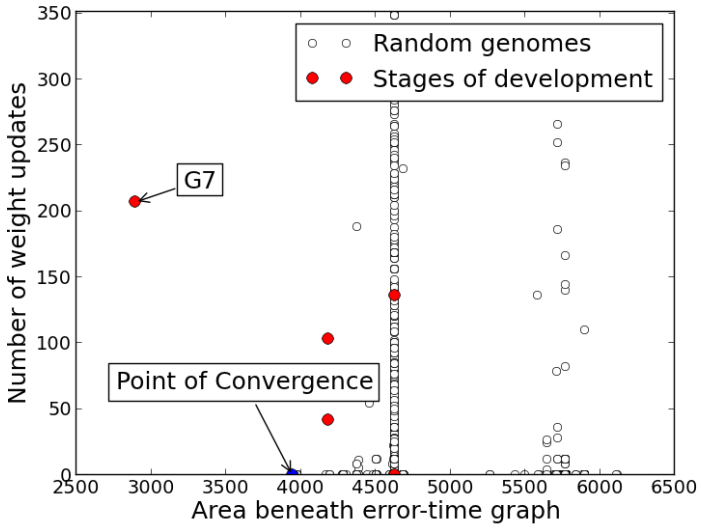
networks, so only 20 trials per logic program were performed.

Some clarification of terms is necessary before proceeding. As with the genomes for relational structures, *group 3* genomes are those which do not produce any connections so that [0,0] ('unknown') is answered for every question. For relational structures in section 3.4.2, *group 2* genomes were those which only formed connections between enablers and collectors of different predicate clusters so that questions were always answered 'true' or 'false' (fig. 4.30a). In the attempt to evolve fact structures, similar genomes were discovered that connected a predicate's enabler directly to one of its collectors so that activating the former always activates the latter (fig. 4.30b). This is equivalent in behaviour to the first axis of development for fact structures (section 4.4.2, figs. 4.28b and 4.28c), even though there are no fact or inhib2 nodes between the enabler and collector. Although the connectivity is different from group 2 networks for relational structures, the behaviour is the same in that all questions asked of a predicate are answered 'true' or 'false'. Therefore such genomes are also labelled 'group 2' for fact structures. In the context of relational structures, *'group 1'* and *'group 0'* both refer to genomes which produce zero-error networks depending on whether or not conjunctive relations are represented. However only 'group 1' will be used to refer to zero-error fact structures that answer all questions correctly.

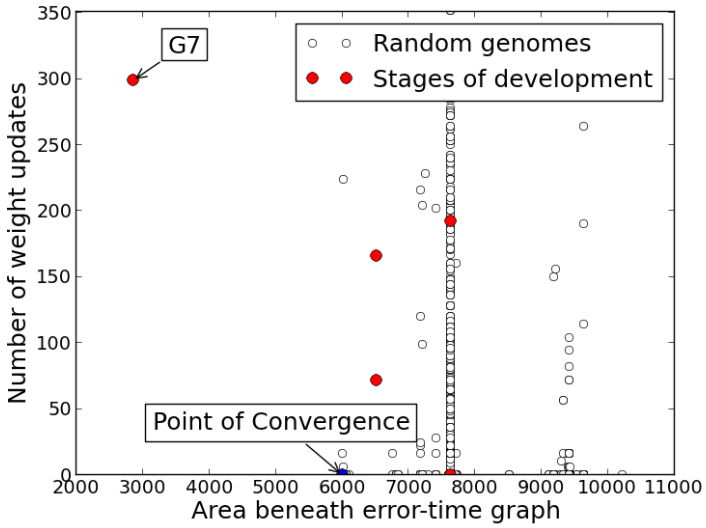
Evolution was attempted for the facts from the NoNeg7 and NoNeg8 logic programs. As with the evolution of mediator structures in section 4.3.5, complete fact structures were not discovered. However although mediator evolution yielded a range of points on the Pareto front, all trials of Fact evolution converged very quickly on one group 2 genome with 0 weight updates (fig 4.31). Reasons for this fast convergence are discussed below.

Fig. 4.32 on page 164 plots the objective values yielded by 50,000 randomly generated genomes for NoNeg8. No group 1 genomes similar to G7 are generated and like the group 0 mediator genomes, are distant from the majority of points in the cloud of objective values produced by random genomes (fig. 4.31). For NoNeg8, group 3 genomes yield an e-area of 7631.[3] Generation of group 3 genomes is even more probable when developing fact circuits (97.18%) than developing mediator structures (91.69%), for the following reasons. The bare minimum required for networks to yield any output is a path between

[3]Note that as was the case in the mediator experiments, group 3 genomes do not necessarily yield the lowest e-area because the error function (equation 4.2, page 140) does not penalise predicates for which all questions are expected to be answered as 'unknown'.



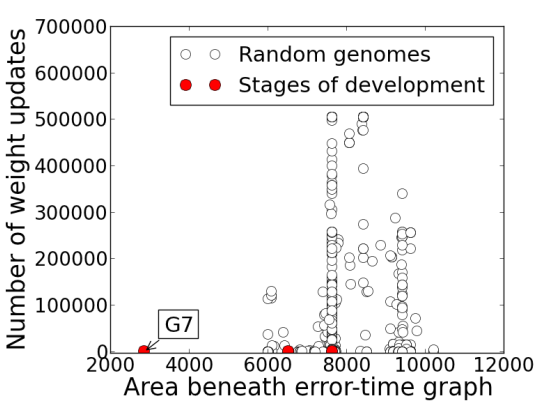
(a) NoNeg7



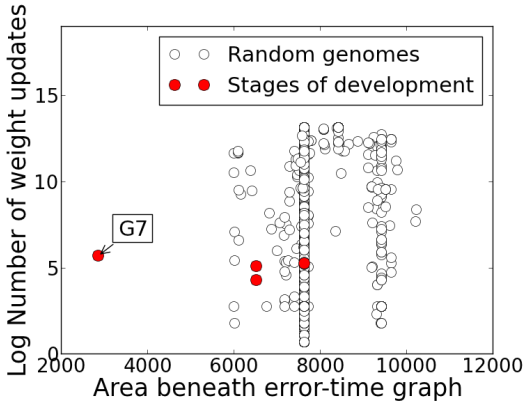
(b) NoNeg8

Figure 4.31. All 20 trials of 500 generations when evolving fact structures converge on a single-point Pareto front. This is compared against the fitnesses of 50,000 randomly generated genomes and fitnesses yielded by each stage of development, generated separately from the evolutionary search.

a predicate's enabler and its collector. If such paths pre-exist before development it is easier to randomly generate genomes which develop networks that yield some output and therefore answer at least some questions (though not necessarily correctly), but with no such pre-existing paths it is more difficult to do this. In the mediator experiments, fact structures already existed before development took place and therefore provided such a path from enablers to collectors so that it was easier for randomly generated networks to answer questions. However now that the fact structures themselves need to be discovered, no such path from enabler to collector pre-exists and it is more likely that random networks will always answer [0,0] (unknown). Therefore it is more likely that group 3 genomes will be randomly generated in fact evolution than in mediator evolution.



(a) Standard scale



(b) Logarithmic scale

Objective pair

	e-area	#Updates	Percentage
1	7631	0	81.10
2	7632	16	11.63
3	9415	0	1.67
4	7631	12	0.75
5	7631	6	0.35
6	7631	3484	0.22
7	7631	10	0.22
8	7631	32	0.11
9	7631	9	0.10
10	9404	0	0.09

e-area

e-area	Percentage
7631	97.18
9415	1.79
9192	0.10
9404	0.10
9327	0.09
8427	0.08
9638	0.08
7719	0.07
9426	0.06
6001	0.05

Number of Updates

#Updates	Percentage
0	83.44
16	11.67
12	0.75
6	0.35
3584	0.22
10	0.22
32	0.11
9	0.10
3	0.09
505344	0.09

(c) 10 most common objectives and objective pairs.

Figure 4.32. Objective pairs yielded by 50,000 randomly generated genomes. The vast majority of genomes are group 3 genomes that yield an e-area of 7631 as a result of answering [0,0] (unknown) for all questions.

The enabler-to-collector path must be discovered in fact evolution before error or e-area can be influenced at all. Until then group 3 genomes with zero weight updates, which is the vast majority of such genomes, will Pareto-dominate all other solutions because zero is the absolute minimum for the second objective. Therefore the initial population contains a Pareto front of only one genome: a group 3 genome with no weight updates. Because group 3 genomes occupy a large portion of the fitness landscape and dominate other early genomes so easily, they are often selected for variation and easily mutated into genomes of the same group. As a consequence they rapidly multiply and fill the population until the population contains no other genomes at all. However because enabler-to-collector paths pre-exist in mediator experiments, it is easier for randomly generated networks to influence error and e-area. It is therefore more likely that the initial Pareto front contains multiple points, and less likely that it will converge on only one point.

Nonetheless, in all cases the one-point, group 3 Pareto front in fact evolution is still able to progress to a group 2 genome, in which the enabler-to-collector path does exist in its simplest form as a direct path between the two nodes (fig. 4.30b). Although this now enables questions to be answered, albeit incorrectly for some questions, the number of weight updates is still zero because the connection is direct and the input (enabler) and output (collector) nodes are activated at the same time. Both learning methods used in

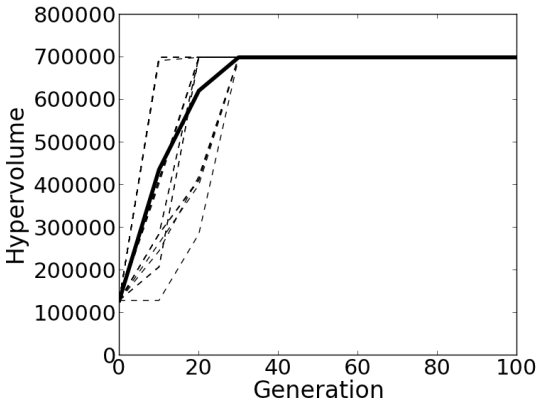


Figure 4.33. The change in hypervolume over 100 generations for all 20 trials. The thick black line represents the average change in hypervolume, which converges in about 30 generations. Therefore the Pareto front also converges quickly when attempting to evolve fact structures.

these experiments (page 49) require that one node is active before another in order for the connection weight to be adjusted. A zero-updates network with a lower e-area has now been discovered and dominates the group 3 genome so that the group 2 genome now assumes the role of the one-point Pareto front. Again, the population quickly converges on this new point for three reasons: it easily dominates other points by virtue of yielding zero weight updates, the genome is difficult to improve through mutation, and the diversity of the population with respect to objective fitness is already very low because the group 3 genome filled previous populations. The fast convergence is indicated by the convergence of the hypervolume in fig. 4.33.

Although a connection from the enabler to the collector is equivalent to the first axis of development for fact structures, a necessary step for the development of fact circuits, progression to the second axis requires inhibitory nodes to lie between the two nodes. Because the path from collector to enabler is functionally equivalent with or without inhibitory nodes, the discovery of these necessary components is not reflected by an improvement in objective fitness.

With such low diversity in objective values on the Pareto front, the evolutionary search becomes equivalent to a random one and the discovery of genomes like G7 that require precise combinations of rules, conditions and actions in order to function becomes highly unlikely. As with mediator structures, the problem is that the fitness landscape is largely populated with group 3 genomes (fig. 4.32), the group 1 genome is distant from more reachable points in objective space (fig. 4.31b), and necessary changes to the genotype are not reflected by the change in objective values or lack thereof. The following set of experiments demonstrate these points further.

4.4.5. Fitness landscape

The same experiments performed on G5 in section 4.3.6 were performed on G7 to explore the fitness landscape surrounding it. The aim of these experiments was to demonstrate that G7 and similar genomes are difficult to discover through evolution because the discoveries

of individual components necessary for the complete fact structure are not always reflected by positive changes in objective fitness. The first experiment isolates rules in the genome, the second one performs controlled mutations and the third performs two iterations of random mutations. All experiments are performed with NoNeg8 (fig. 4.8, page 128).

Inhibitory experiments

These experiments isolate rules in the genome so that the performance of other rules can be observed in isolation and in different combinations. Each possible combination of rules is explored in table 4.12 and fig. 4.34 plots each pair of objective values obtained. As explained in section 4.4.2, the development of each axis enables the development of the next. The results in table 4.12 and fig. 4.34 show how objective values change as each rule is executed and as each axis develops:

Before development: No connections are developed and all questions are answered as 'unknown', yielding an e-area of 7631. This is equivalent to a group 3 genome.

Axis 1 - Creating fact nodes (rules 1 and 2): A fact node between the enabler and the collector is formed by rule 1. Because of the uninhibited connection between these two nodes, questions on predicates that have facts associated with them are always answered 'true'. These are equivalent in behaviour to group 2 genomes and yield an e-area of 6509.

Axis 2 - Connecting predicate arguments (rule 3): Inhibitory connections from role nodes are formed. However, because these inhibitory nodes are not themselves inhibited, they will always inhibit the activation of the fact node when the predicate is queried, returning the network to a state that always answers 'unknown' and yields an e-area of 7931, just as it did before development began.

Axis 3 - Connecting entity nodes (rule 4): Connections from entity nodes are formed which inhibit the level 1 inhibitory connections from role nodes when argument bindings are correct, and all questions can now be answered correctly. These are equivalent to group 1 genomes and yield an e-area of 2851.

In order to produce the full, working genome, the rules for constructing each of these axes must be discovered in order, and in the process, the error increases upon discovering the second axis instead of decreasing in the direction of zero-error (rule 3, fig. 4.34a). From fig. 4.34b it can be seen that genomes produced upon the discovery of rules 2 and 3 are Pareto-dominated by the other genomes and therefore the steps that must be taken in genotypic space do not reflect progress towards the target behaviour in objective space. This is typical behaviour for generative and developmental systems and makes rediscovery of the target genome difficult [122].

Any rule combinations other than those that follow this sequence yield the same objective values as the largest subset of its rules that form the correct sequence. For example, the rule combination 1,2,4 yields the same e-area and number of updates as the rule combination 1,2, because rule 4 cannot be expressed until rule 3 has been expressed. Any

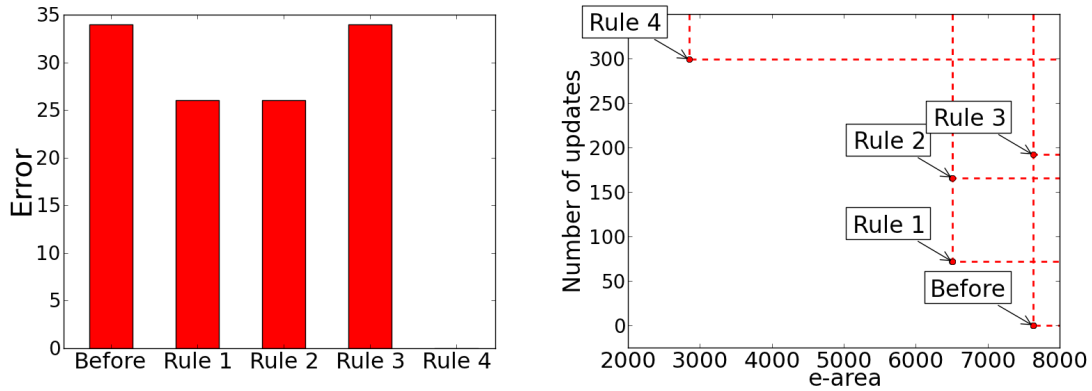
Rule combinations

Rule	Error	e-area	#Updates
1	26	6509	72
2	34	7631	0
3	34	7631	0
4	34	7631	0
1, 2	26	6509	166
1, 3	26	6509	72
1, 4	26	6509	72
2, 3	34	7631	0
2, 4	34	7631	0
3, 4	34	7631	0
1, 2, 3	34	7631	192
1, 2, 4	26	6509	166
1, 3, 4	26	6509	72
2, 3, 4	26	6509	0
1, 2, 3, 4	0	2851	299

Development of axes

Rule	Error	e-area	#Updates
None	34	7631	0
1	26	6509	72
1, 2	26	6509	166
1, 2, 3	34	7631	192
1, 2, 3, 4	0	2851	299

Table 4.12. Objective values obtained when discovering different combinations of rules for developing fact structures for the NoNeg8 logic program, for which the maximum error is 42. The error yielded by group 3 networks is 34.



(a) Error

(b) Objective values.

Figure 4.34. The change in objective values as each rule of fact development is discovered. Plotted values are taken from table 4.12

rule combinations that exclude rule 1 yield the same values as a genome that contains no rules, because all other rules require rule 1 to have been expressed first. In order for the discovery of any rule to be relevant, the rules preceding it in the developmental process must also have been discovered.

In summary, once again we see that the individual rules in the genome that are necessary for constructing target networks do not necessarily improve objective fitness when discovered individually, making it difficult for target genomes to be discovered.

Controlled mutation

The landscape surrounding genome G7 was then explored by mutating condition arguments along the range of possible values. Mutation of activation-based conditions such as those testing time window and phase are excluded for brevity. Once again, for all conditions, only particular values yield an error of zero and the corresponding e-area. Therefore only a precise combination of condition values can produce zero-error networks.

Fig. 4.35 presents the results of mutating conditions pertaining to node function. The results are similar to those from the controlled mutations of mediator genomes (fig. 4.23, page 150) in that the resulting objective values are identical to those produced from the inhibitory experiments because mutating a condition is equivalent to inhibiting the rule it participates in. Genomes formed by inhibiting rules are equivalent to group 2 or 3 genomes in that they either always answer 'true' or always answer 'unknown' for all questions.

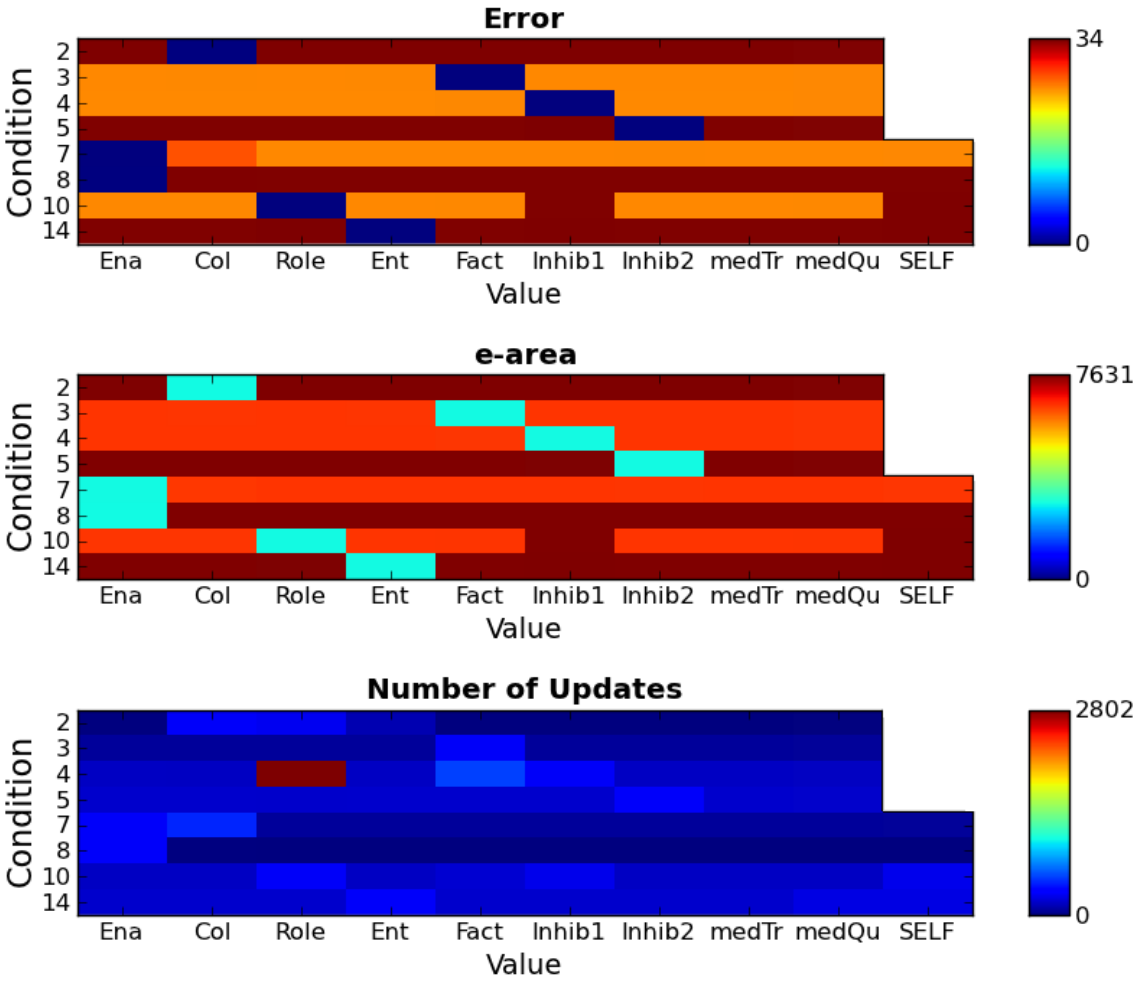


Figure 4.35. Controlled mutation of conditions on node function. The y-axis represents conditions from G7 (fig. 4.27) that pertain to node function, and the x-axis represents values that the arguments of those conditions may take ($SELF.Function = Ena$, $SELF.Function = Col$, etc.). The colour of each square represents the objective value yielded by the network that develops when the argument of the condition in the y-axis is set to the value in the x-axis. Conditions 2-5 are conditions on SELF (e.g. $SELF.Function = X$) and conditions 7, 8, 10 and 14 are conditions on E_CON or P_CON (e.g. $P\_CON.Function = X$). Conditions on E_CON and P_CON may take SELF as an argument ($P\_CON.Function = SELF.Function$), but conditions on SELF may not, hence why no results for SELF as an argument are provided for conditions 2-5.

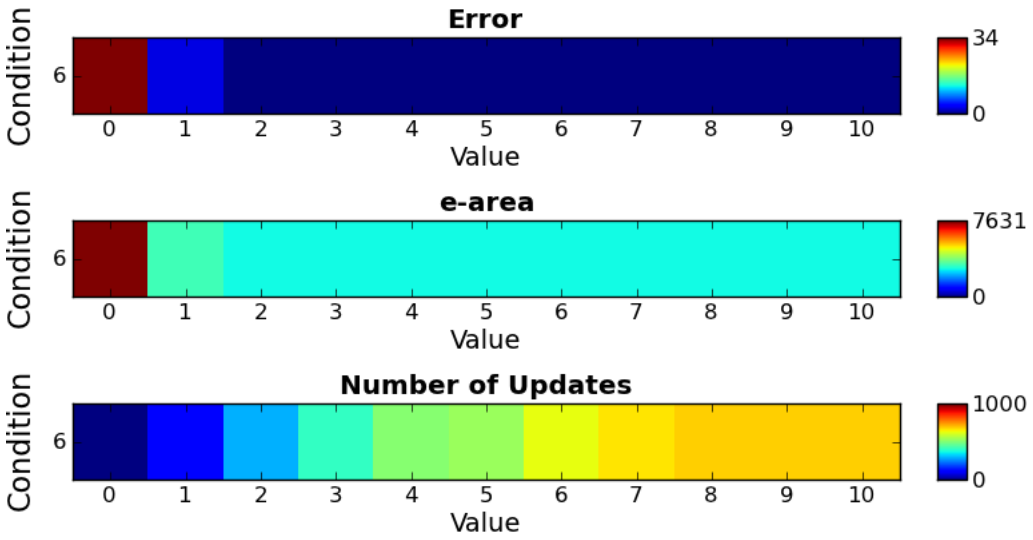


Figure 4.36. Controlled mutation of condition 6 from G7, which tests for the number of inputs to a node (nInputs). The x-axis represents values that the argument of condition 6 may take ($SELF.nInputs = 0$, $SELF.nInputs = 1$, etc.).

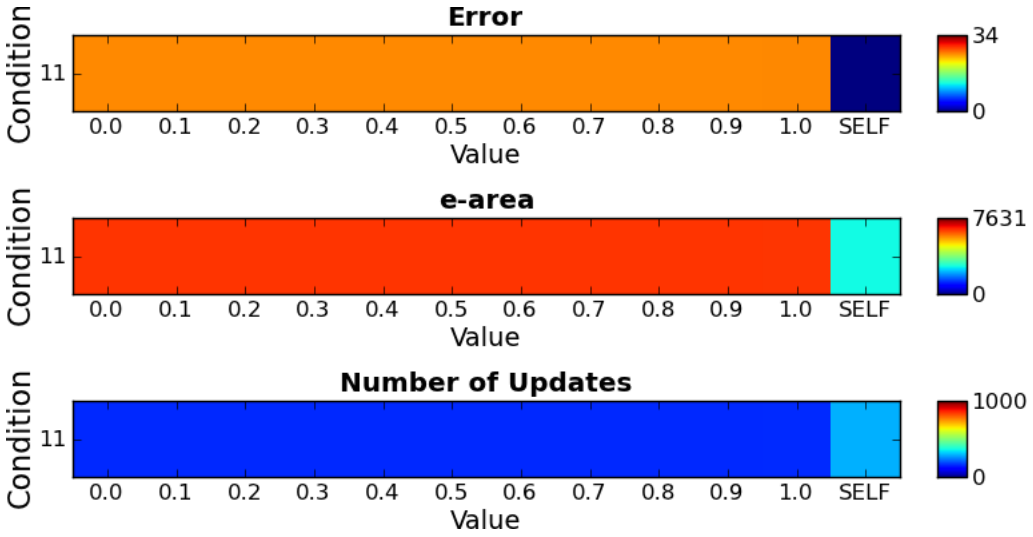


Figure 4.37. Controlled mutation of condition 11 from G7, which tests for the amount of chemical 2 held by a node (chem2). The x-axis represents values that the argument of condition 11 may take ($P\_CON.chem2 = 0.0$, $P\_CON.chem2 = 0.1$, etc.).

Mutating the number of inputs (fig. 4.36) is only problematic when set to a value lower than the maximum number of facts associated with any predicate (two). In NoNeg8, for which these genomes were developing fact structures, predicates P and T each have two facts associated with them, and therefore limiting the number of inputs to each predicate cluster's collector to a value less than 2 prevents the required number of facts from developing. The number of updates does not increase for values of $nInputs$ greater than 8, because by the time this many weight updates have been made, the learning-development cycle has finished.

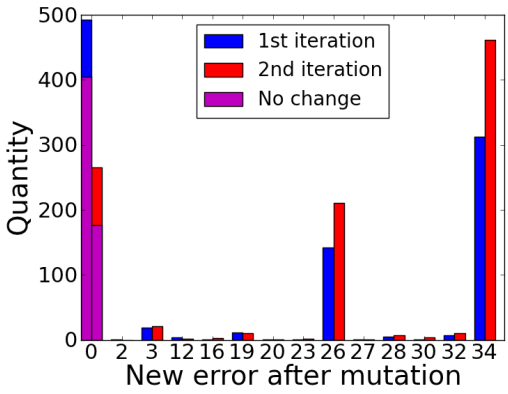
Mutating chemical 2 (fig. 4.37) to anything other than 'SELF' prevents the correct connections from being formed between between role nodes and inhib2 nodes, and therefore only the first axis develops correctly.

These experiments show that the fact genome requires not only a precise combination of

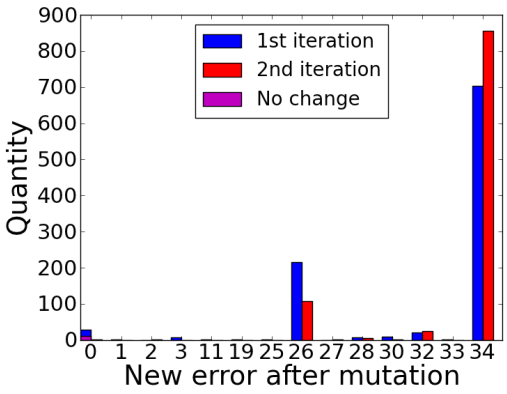
rules but also a precise set of conditions in those rules in order for fully functional fact structures to be produced. Furthermore, the fitness landscape surrounding these genomes is largely populated by group 2 and 3 genomes as a result of certain rules being inhibited by the mutation of their conditions.

Random mutation

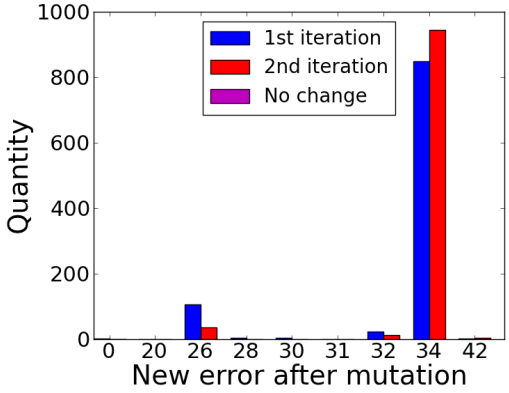
In this final set of experiments, 1000 copies of genome G7 were mutated once to demonstrate how many genomes in the surrounding landscape are group 3 genomes, and a second time to demonstrate how unlikely it is that the child genomes produced by the first iteration can be mutated back into group 1 genomes that produce networks that can answer all



(a) pMut = 0.01



(b) pMut = 0.05



(c) pMut = 0.1

Figure 4.38. Error values produced by random mutation of 1000 copies of the target genome G7 according to different mutation rates. Two iterations of mutation were performed on each copy of the genome. The majority of mutations result in group 3 genomes that always answer [0,0] (unknown) and yield an error of 34. The maximum possible error is 42.

	Number of reductions in error	
Mutation rate	**Any**	**To zero-error**
0.01	26	0
0.05	48	0
0.1	30	0

Table 4.13. The number of reductions in error made by each mutation rate in the second iteration of random mutation. 1000 genomes were mutated for each mutation rate

questions correctly. The fact genome was randomly mutated according to three mutation rates: 0.01, 0.05 and 0.1. In a genome of 30 conditions or actions, each mutation rate mutates approximately 0.62, 3.09 and 6.12 exons respectively. Results are presented in fig. 4.38. Once again, most of the error rates yielded correspond to those found from inhibiting rules in the genome, thus supporting the hypothesis that the landscape surrounding the target genome is largely composed of group 3 genomes that yield an error of 34. Table 4.13 shows how many networks have their error values improved in the second iteration of mutation, and on no occasions were genomes mutated back to those that could produce group 1 zero-error networks. As with the mediator structures, it is easy to mutate a zero-error genome into a group 3 genome, but much less likely that a group 3 genome will mutate into a zero-error genome. This is even the case when mutating a group 3 genome that is close to a zero-error genome in genotype space. Therefore the fitness landscape does not contain the necessary information to indicate proximity to fitter genomes.

In summary, the results of attempting to evolve fact structures through were very much the same as they were for mediator structures, and for the same reasons. The structure can be represented in a genotype, but not easily rediscovered through evolution due to the brittle nature of SHRUTI networks. A large number of randomly generated genomes and mutations of other genomes produce networks that always answer [0,0] ('unknown') for any questions they are presented with and necessary changes to the genotype are not indicated by the objective fitness values they yield. The algorithm quickly converges on group 2 genomes, which are functionally equivalent to the first axis of fact structure development. Because it is difficult to improve upon this or other genomes in a way that is reflected by the resulting objective fitness, evolution is highly unlikely to discover anything better.

Unlike the evolutionary search for mediator genomes, the evolutionary search for fact genomes was not given the freedom to discover genomes that do not limit development by activation. This choice was made because the networks that such genomes would produce would be too large to execute in a reasonable amount of time (section 4.4.2). Like the evolutionary search in chapter 3, the search for mediator genomes was allowed the freedom to discover genomes not constrained by activity as a precursor to more efficient genomes that were constrained by activity. Therefore the inability of the evolutionary search to discover fact genomes may not be supported as strongly as it was for the mediator genomes. However, even if the fact genomes were not constrained by activation, it is highly likely that they would be just as difficult to discover in the evolutionary search. All experiments exploring the fitness landscape involved the mutation of conditions unrelated to activation, and group 2 or 3 genomes were produced in most cases.

The SHRUTI fact model for which these arguments have been made is a rather simple one, especially since it does not include a means of unlearning facts and a much more intricate alternative exists in the form of SMRITI [91, 94]. However, considering that the evolutionary search struggled to discover a genotype representation for a simple fact model, it is highly unlikely that it would be able to discover a genotype representation for any improved version of that model, including SMRITI.

4.5. Discussion

In this chapter, the genome model was improved to enable the development of new nodes in addition to the connections between them. This was necessary to enable the development of mediator and fact structures. Scalable indirect encodings for mediator and fact structures have been presented and shown to develop structures capable of representing the training data they are presented with and correctly answering a set of test questions. However, as was discovered in section 3.2.4, it is difficult to constrain the size of the phenotype without limiting the development of connections and intermediary neurons to those between active neurons because in order for a fact or relation to be learned, its corresponding structure must exist for it to be learned in the first place. This is a consequence of the localist representation employed by SHRUTI, as was argued in section 3.6. Although neurogenesis does involve some activity-dependent development [33], the way it is applied in G6 lacks biological plausibility because it makes some aspects of the learning algorithm redundant and because learning influences the developmental process throughout. In biology, some development occurs in the pre-natal stage before any learning can take place. Nonetheless, regardless of whether or not development is triggered by activation, the description of SHRUTI's mediator and fact structures have been successfully encoded in the genome, supporting the claim of SHRUTI's developers that SHRUTI can be represented using indirect encoding [96]. Because indirect encoding is a biologically plausible encoding method, the biological plausibility of SHRUTI networks has been supported to some extent by showing that they can be encoded in this way. Furthermore, the updated genome model has gained some new aspects of biological plausibility in addition to those highlighted in section 3.2.4. In particular, neurons may now be produced through development in addition to connections [1, 12, 123], and chemical concentration levels may be used to interpret positional information that affects development [116]. However as argued in section 3.2.4, biological plausibility is still limited in the sense that the model is an abstraction of the much more complex process of real biological development.

The NSGA-II algorithm was unable to discover mediator or fact genomes in an evolutionary search, but an examination of the target genomes and the fitness landscape were informative as to why this was the case and what future steps might improve evolvability. In chapter 3, genomes were easily discovered due to the simplicity of the basic requirements for representing a relational structure that does not require mediator nodes. Only one condition testing for function equality was required, and as soon as this was discovered, the evolutionary search could work on optimising the efficiency of the genome. However, genomes capable of developing mediator and fact structures were much more complex, and both required a number of rules, conditions and actions to be discovered together in precise combination. Individual discovery of these components make the correct steps in genotype space but this is not reflected by the resulting change (or lack thereof) in objective fitness. This sort of misdirection is a common phenomenon in other generative and developmental systems [122]. Furthermore, a large majority of randomly generated genomes and mutations of existing genomes produce group 3 genomes that always answer 'unknown' by never firing any collectors. In general, the fitness landscape contains insuf-

ficient information to identify necessary building blocks for constructing fitter genomes in later generations and therefore any evolutionary algorithm would struggle to navigate it.

The evolutionary algorithm takes the essential building blocks of SHRUTI (nodes and connections as specified in the literature), and attempts to rearrange them into a working network structure. In this respect the description of SHRUTI structures made by the genome is quite explicit, suggesting that the difficulties with evolving SHRUTI structures are not owed to the genome model but to the structure of SHRUTI networks themselves. SHRUTI behaves in a very discrete way. The performance of SHRUTI ultimately depends on the existence of relational and fact structures, which affects which questions are answered, and the existence of these structures is a discrete factor owed to the localist representation employed by SHRUTI networks. The results demonstrate that SHRUTI networks do not degrade gracefully, in that each individual component of a SHRUTI structure is required in order for that structure to function properly. A lack of graceful degradation is a common criticism of localist and symbolic models [77]. In order to improve evolvability, the idea of analysing the structure of a network and rewarding particular substructures that are known to improve performance was considered. However, this is very biologically implausible and points to a much more directed process than evolution is known to be. The only alternative would be to somehow measure the emergence of these substructures at the behavioural level so that their emergence is reflected in existing or new objective values, but it remains to be seen how this could be achieved.

One improvement that would support, but not necessarily solve, the search for SHRUTI networks would be to somehow smooth the fitness landscape and make it more continuous in such a way that small genotypic discoveries yield more informative changes in objective fitness. The experiments in this thesis were conducted on a simple SHRUTI model that uses one neuron per concept, and the use of ensembles containing multiple neurons per concept may smooth the fitness landscape somewhat by testing for activation of each member of each ensemble. However, evolution would probably still fail because the model still uses localist representation and the same discoveries such as function equality and axis interaction still have to be made by the evolutionary algorithm. Furthermore, e-area is not continuous but only granular since it is a function of error, which is discrete. Error is discrete because a collector's activation is also discrete, either 0 or 1. Activation could be made continuous by setting the output of a collector as a function of its input potential, which is continuous. However the problem still remains that between the discovery of genomes of each group, conditions and rules must be discovered that do not affect the output of the collectors when discovered individually. Even if the output is continuous, the transition between stages of development remains discrete. In order to make it more likely that SHRUTI or any other neural-symbolic network can be discovered through evolution, the network's output needs to be continuous in the sense that the discovery of each substructure and the discovery of each rule, condition and action that contributes to each of those structures provides some individual contribution to objective fitness in the direction of the target behaviour.

The idea of smoothing the transition between different genome groups and stages of de-

velopment has some biological plausibility. In section 3.4.2 on page 108 it is argued that transition between discrete stages of development as argued by Piaget [78] is a view no longer held by modern psychologists, who regard the transition between stages as being more continuous [100]. Difficulties encountered in discovering transitions between discrete stages in this chapter supports this theory.

It has been argued in section 2.1.4 and in the literature that it is unlikely that the brain uses localist representation, and that it is more likely that it uses distributed representation [80]. Even if the brain is localist to some extent, even proponents of the localist hypothesis agree that a localist representation may be partially distributed as long as any one neuron encodes for one concept more than it does for any others [77]. The difficulty of discovering fully localist[4] representations of neural-symbolic reasoning such as SHRUTI through evolution supports this hypothesis even further because it suggests that the discovery of fully localist neural representations through natural evolution may also be unlikely. These observations coupled with the fact that localist representations also make it difficult to constrain network size without copying aspects of the learning algorithm supports the argument put forward in section 3.2.4 that exploration of a new, more distributed and continuous neural-symbolic model of reasoning would be worthwhile. Rather than development of new connections being triggered by observations of the relations they represent, development could instead occur as greater demand for storage capacity or other performance factors are required, similar to the information theory of cognitive development [69, 100]. A distributed model based on the information processing theory is also consistent with the idea of more continuous and less discrete transition between stages, as information processing theory is among those theories that support such transition [69, 100]. For the purposes of evolvability, a more distributed and continuous representation may allow individual discoveries to make their own contributions to objective fitness so that the fitness landscape becomes more informative. A distributed representation does not commit any neuron or connection to any particular concept or function because a many-to-many relationship is used. Each neuron participates in the representation of multiple concepts or functions and each concept or function is represented by multiple neurons. Therefore if, for example, a connection between role nodes is discovered, and those role nodes also contribute somehow to the activation of enabler or collector nodes, this discovery would provide some contribution to the output of the network. However in the original, localist SHRUTI, role nodes are only role nodes and cannot influence the output of the network until enabler-to-enabler and collector-to-collector connections have been discovered.

In general, successful implementation of a more distributed SHRUTI would improve its biological plausibility by demonstrating that it is compatible with both localist and distributed theories of representation. However, it is difficult to know whether or not this would actually solve the problem of evolvability until the model has actually been designed and implemented. The following chapter explores possible directions that could be taken in producing distributed and continuous neural-symbolic reasoning systems.

[4] The reader is reminded that the term 'fully localist' is used to refer to localist representations which are not even partially distributed

5. Distributed SHRUTI Networks

This chapter serves three purposes: to argue a case for exploring a distributed SHRUTI model, to suggest future steps that should be taken in doing so, and to present the results of some initial experiments that highlight some difficulties that will need to be overcome.

The current SHRUTI model uses fully localist representation in that there is a one-to-one relationship between neurons and the concepts they represent, or many-to-one in the case of ensembles. This is likely to be one of the reasons that SHRUTI mediator and fact circuits were difficult to discover through evolution in chapter 4. In both cases, individual discoveries necessary for producing the structures may yield no changes in objective fitness or may yield some change but in the incorrect direction in the fitness landscape. If each discovery provides some individual contribution to objective fitness, then this may help to smooth the fitness landscape and provide it with more information to guide the evolutionary search. Such a behaviour may be possible through distributed encodings. Whether a model is fully distributed or localist with partial distribution, neurons may encode some portion of a number of concepts or participate in a number of functions. In a partially distributed localist model, a neuron would encode mostly for one function or concept but also have some participation in the encoding of another. For example, a neuron might mostly participate in the encoding of one predicate argument but also provide support for another. Another neuron may participate mostly in the propagation of role bindings but also provide some support in propagating enabler signals. In chapter 4, discovering role node connections alone yielded no contribution to objective fitness on its own. However discovering role node connections in this hypothetical distributed model may also provide some activation to enablers and collectors and therefore make some contribution to objective fitness.

Another reason to explore more distributed encodings is that localist representations as used by SHRUTI are largely thought to be less biologically plausible [80]. Whilst localist models make important contributions to cognitive modelling [77], distributed or partially distributed models may provide some understanding as to how reasoning is represented and processed at the neural level. Neuroscience largely favours the distributed theory of representation and even localists would argue that localist encodings could be at least partially distributed [77]. In the current SHRUTI model, this is not the case. The one-to-one or many-to-one relationship between neurons and concepts is fully localist in that any neuron only encodes for one concept with absolutely no contribution to any other. A model of distributed encoding for SHRUTI would support the biological plausibility of SHRUTI by demonstrating that the neurally plausible binding mechanism that SHRUTI contributes to cognitive modelling is compatible with both distributed and localist theories.

Furthermore, it was argued in chapters 3 and 4 that the localist representation method used in SHRUTI meant that in order for any relation to be learnable, its neural representation had to either pre-exist or be constructed upon its observation. In the former case, development becomes redundant after constructing all possible relations. In the latter case, learning is redundant because development has already produced the structures to be learned and no others. It is more likely that development supports learning in a way that makes neither process redundant. In a more distributed representation, development could support learning in a way similar to the information processing theory of cognitive development [69, 100] by improving factors such as storage capacity and connectivity in order to improve performance rather than building a representation of what needs to be learned. For example, development could provide new connections when the system becomes saturated with knowledge or error becomes high. A distributed model based on information theory is also consistent with the idea of smoothing the transition between discrete stages that made it so difficult to evolve SHRUTI networks. Information theory, among other modern theories of development [100], supports a more continuous and therefore less discrete transition between developmental stages than Piaget suggested [78].

In summary, the reasons for exploring the possibility of a distributed model of SHRUTI are two-fold: biological plausibility and possible improvements to evolvability, which in itself would contribute to the biological plausibility of SHRUTI. It is unreasonable to attempt the evolution of a distributed SHRUTI model until the implications of such a model have been explored in detail. This chapter provides an initial discussion on this matter and suggests what steps should be taken in order to work towards a distributed model. The following summarises the traits that a distributed model of SHRUTI should exhibit:

- Many-to-many relationships so that a neuron may participate in the representation of multiple concepts (e.g. entities and predicate arguments) or functions (e.g. role nodes, enablers and collectors) and each concept or function is represented by multiple neurons.

- The ability to degrade gracefully so that the loss of one neuron participating in a representation does not result in that representation being lost from memory.

Like other fully localist neural-symbolic reasoners, SHRUTI can be regarded as an extension to the general relational structure (section 2.3.3) in which each antecedent, consequent and rule is represented by an individual neuron or set of neurons. Therefore, the task of distributing representations in SHRUTI and evolving the structure extends to neural-symbolic reasoners in general. SHRUTI's unique contribution to the general relational structure is the use of spiking neurons and temporal synchrony for dynamic variable binding. Distributed representations of neural-symbolic reasoners do in fact already exist [3, 6, 46, 48, 102, 109], so the next logical step to take in enabling distributed reasoning in SHRUTI is to explore the possibility of combining SHRUTI's temporal binding capabilities with distributed representations. If the union of SHRUTI's binding mechanisms and distributed reasoners is successful, then distributed alternatives to other SHRUTI representations such as facts and type hierarchies can then be considered. In summary, there are three stages to exploring the possibility of distributing SHRUTI networks:

1. Existing distributed neural-symbolic reasoners.

2. Distributed neural-symbolic reasoners integrated with SHRUTI's spiking neurons and temporal binding mechanism.

3. Spiking, distributed neural-symbolic reasoners integrated with distributed mechanisms for further SHRUTI representations such as facts and type hierarchies.

It would also be reasonable to assess the evolvability of each of these stages individually. For example, if difficulties are encountered when attempting to evolve existing distributed reasoners, it may not be worthwhile attempting to evolve the following stages. If distributed SHRUTI networks are possible but not evolvable, at the very least a step towards biological plausibility will have been made by demonstrating that variable binding by temporal synchrony is compatible with both localist and distributed theories of representation.

The following sections expand on the points made above in order to argue a case for exploring a distributed SHRUTI representation and suggest future directions that should be taken, complete with some preliminary results. First, section 5.1 elaborates on the point that because all fully localist neural-symbolic reasoners adhere to the general relational structure, any solution to distributing SHRUTI relations may also apply to other localist models and vice-versa. Existing distributed neural-symbolic reasoners are then reviewed in section 5.2 to explain why integration with associative memories are a logical step to take in exploring the possibility of a distributed SHRUTI model. Section 5.3 presents the preliminary results obtained from combining the principles of an associative memory with SHRUTI's relational structure and shows that in such a distributed structure it is difficult to propagate variable bindings from one cluster to another in such a way that it is possible to identify which predicate argument is bound to which value. This problem must be overcome if the idea of a distributed SHRUTI model is to be pursued any further. Section 5.3 also highlights that should a solution to this problem be found, the variable-binding mechanism may be transferable to CILP++ [34] and the RTRBM [27, 28]. In the latter case, the solution would provide a means of performing probabilistic reasoning on first-order relations. Section 5.4 provides a discussion of a possible alternative approach with regards to distributed neural-symbolic reasoners, inspired by the neuroscientific theories that reasoning is performed by the same cognitive mechanisms that perform visual and linguistic tasks [81]. Section 5.5 summarises.

5.1. Generalising the problem to all localist models

The reasoning components of all fully localist neural-symbolic networks follow the same basic principle of what is referred to as the *general relational structure* in section 2.3.3. Individual antecedents, consequents and rules are represented by individual neurons, and the connections between those neurons represent the relations between antecedents and consequents. For example, consider the relation $A \wedge B \rightarrow C$. In connectionist networks, A, B and C are represented by neurons in the input and output layers, with a hidden node to implement the rule by acting as a mediator between antecedent neurons (A and B) and the

consequent neuron (C). In SHRUTI, A, B and C are represented by individual predicate clusters, with a mediator cluster implementing the rule by propagating activation between the antecedents and the consequent. In all cases, antecedents, consequents and rules are represented by neurons in a one-to-one or many-to-one relationship.

A genome for any localist neural-symbolic reasoning system must contain the concept of the general relational structure and reproduce it multiple times. The existence of relational connections or intermediary nodes necessary to implement this structure is a discrete factor because they either exist or they do not. Therefore, even if outputs of nodes and the measurement of error operate in a continuous domain, the discovery of a relational structure is not continuous in the fitness landscape because the relational structure would have to be discovered as a whole before it can influence the output. This was shown for SHRUTI networks in chapter 4. As has already been argued, the representation of reasoning in SHRUTI follows the same basic principles as all other localist neural-symbolic reasoners, and therefore it is possible that the same difficulties would be encountered in evolving them that were encountered when evolving SHRUTI. However, the implementation of the general relational structure in connectionist networks is simpler since the logic is propositional and each antecedent, consequent and rule requires only one node to represent it, unlike SHRUTI which requires a collector, enabler and role nodes for each. Therefore discovery of the general relational structure need only occur once for connectionist networks, which is much simpler than having to discover it separately for enablers, collectors and role nodes as in chapter 4. Similarly, in chapter 3, the relational structure needed to be discovered only once because the $P\_INPUT.Function = SELF.Function$ condition could be applied to all node functions. This was indeed easier for the evolutionary search to discover than the conjunctive relational structures.

Regardless of whether or not other localist reasoning models are evolvable, the issues observed in SHRUTI networks in chapters 3 and 4 regarding prerequisites for learning relations also apply. In all the localist models, every possible relation can be represented by constructing every possible connection between antecedent and consequent nodes. If all connections are initially weak, a learning algorithm could strengthen connections corresponding to the desired relations. However to guarantee that all desired relations can be learned, the representation of those relations must exist to be learned in the first place. As was argued for SHRUTI, the only way to ensure that a connection or mediator node exists is for it to pre-exist or for the developmental process to construct it when it is needed, which makes the learning algorithm redundant. With a distributed alternative to the general relational structure, this may not be the case. Development could involve the production of new nodes or connections in order to improve storage capacity or processing power, akin to the information processing theory of development [69, 100]. Development would therefore support learning in such a way that makes neither process redundant.

This hypothetical distributed alternative to the general relational structure is worth exploring because of possible improvements to biological plausibility by virtue of similarities with the information processing theory and because it would demonstrate that the idea of a general relational structure is compatible with both localist and distributed theories

of representation. Because localist neural-symbolic reasoning systems have the general relational structure in common, a distributed alternative to one may be adaptable to the others. Existing distributed reasoning models are reviewed in the following section.

5.2. Distributed neural-symbolic reasoning systems

This section considers existing distributed neural-symbolic reasoning systems and their suitability for integration with the SHRUTI model. Of course, this does not necessarily resolve the issue of evolvability for SHRUTI or for any neural-symbolic reasoners, but it won't be possible to test evolvability until a suitable distributed model has been chosen.

INFERNET [102] and LISA [48] use distributed representations and like SHRUTI, employ variable binding using spiking neurons and Hebbian learning. LISA is partially distributed and performs reasoning by analogy instead of using a reasoning model similar to the general relational structure. However inter-node connections that represent analogies and therefore enable reasoning are still localist. Only concept representations are distributed. In order to fully explore the possibility of distributed reasoning systems, networks in which logical relations are also distributed are of interest. INFERNET also only distributes the representation of objects. All long-term knowledge, including relational structures, is localist. Furthermore, the only learning that takes place is the learning of bindings between object clusters. Long-term knowledge cannot be learned and must be hard-coded. Therefore only the binding mechanism is subject to learning, development and evolution in the way that SHRUTI networks are, but not the reasoning system itself. Successful evolution of this distributed binding mechanism does not point towards successful evolution of relational structures.

First-order core networks involve distributed representations and in doing so provide a means of binding variables [6, 46]. However the learning mechanism used is backpropagation as opposed to the more biologically plausible Hebbian learning. Furthermore, first-order core networks require a process of translating ground terms to and from floating point numbers at the input and output stages respectively. Associative memory methods also distribute reasoning over multiple connections and have been shown to be compatible with Hebbian learning [3, 108, 109, 121]. Furthermore, associative memory methods could be claimed to be a distributed extension to the general relational structure because an input layer representing antecedents propagates to an output layer representing consequents, possibly via a hidden layer representing rules. It is therefore possible that the relational structures in SHRUTI can be made more distributed by integrating them with associative memories. Suitable steps for this research to take are as follows:

1. Adapt associative memories to use spiking neurons and variable binding so that bindings can be propagated from one set of neurons to another in an unambiguous way.

2. Adapt this model so that SHRUTI node function (role nodes, collectors and enablers etc) can also be distributed.

3. Modify the representation of SHRUTI facts (and other structures) to be compatible.

Evolution of associative networks at each of these stages should also be performed. If evolution proves difficult at one stage, it would not be worth moving on to the next and may disprove the hypothesis that difficulties in evolving SHRUTI networks are owed to their localist nature. Nonetheless, distributed models at any of these stages would further the biological plausibility of neural-symbolic reasoning systems and would still be relevant for this reason. The following section highlights some difficulties that will need to be overcome with respect to integrating SHRUTI's relations with associative memories.

5.3. Distributed SHRUTI: preliminary results

The general relational structure can be adapted to a distributed alternative by using associative memories in which neurons between layers are all interconnected and weights reflect associations between those neurons. This section explores the possibility of incorporating SHRUTI's dynamic binding mechanism into this idea by representing SHRUTI relations as associative networks. The aim is to explore whether or not role bindings can be propagated unambiguously from layer to layer.

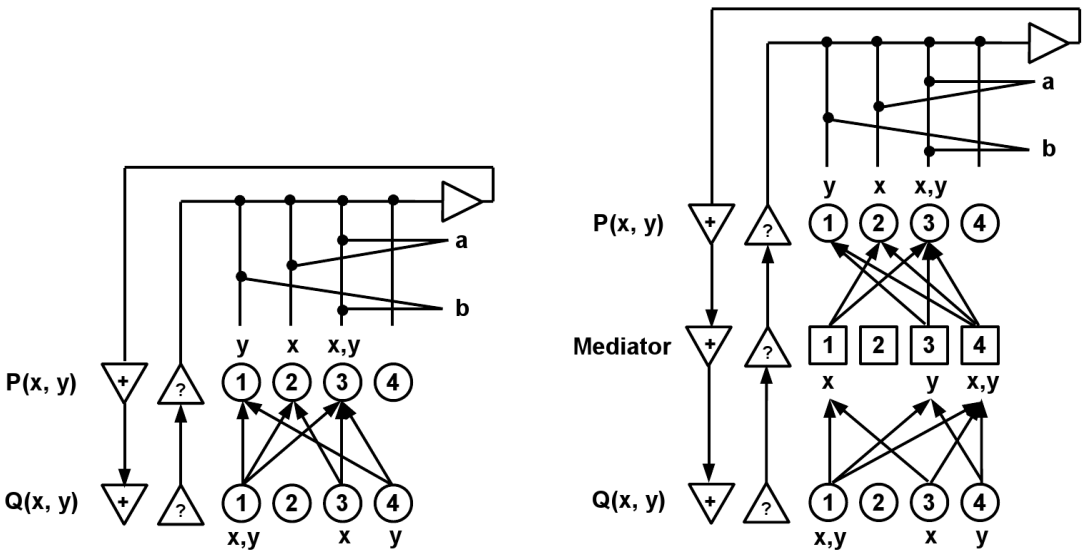
5.3.1. Distributed relational structure

To simplify the problem, distribution among role nodes of the same predicates, but not between role nodes of different predicates, was attempted. Any role node within a predicate cluster may encode for one or more predicate arguments, and any argument may be encoded by one or more role nodes. If bindings could be successfully transmitted from one layer to another so that it was still possible to determine which predicate argument was bound to which entity, only then would a structure involving distribution among different predicates be explored.

In this model, a predicate or mediator structure is represented by n role nodes, and each argument may be represented by any b of these as long as no other argument is represented by the same b nodes. A node in the antecedent or mediator cluster receives input from any node in the previous cluster which shares an argument representation and fires when at least θ of its inputs are active. Figure 5.1a on the next page shows a distributed representation of $P(x, y) \rightarrow Q(x, y)$ with $n = 4$, $b = 2$ and $\theta = 2$. Each argument is represented by two nodes and the third role node of P participates in the representation of two arguments, as does the first role node of Q. Figure 5.1b demonstrates how this circuit might appear if it included a mediator structure.

The truth of a predicate instance is determined as before. A predicate is instantiated by firing role nodes in phase with any entity nodes to which arguments represented by those nodes are bound[1]. The predicate instance is propagated through connections between

[1]The reader is reminded that role nodes (and other m-ρ nodes) may fire in multiple phases so that the predicate arguments they represent may be bound to multiple entities.



(a) Distributed representations of predicate arguments without mediator structure.

(b) Distributed representations of predicate arguments with mediator structure.

Figure 5.1. Distributed representations in SHRUTI networks based on associative memories. Representations are distributed within predicate clusters so that any predicate argument may be represented by multiple neurons, and any neuron may participate in the representation of multiple arguments.

predicate nodes and dynamic bindings are matched against static bindings encoded in fact circuits. Any bindings that do not match inhibit the activation of the fact node to indicate that the fact neither confirms or denies the truth of the current predicate instance.

Returning to the example in figure 5.1a, to ask the question $Q(a, b)$?, x is bound to a and y is bound to b. Therefore, any node that participates in the representation of x must fire in phase with the a entity node, and any node participating in the representation of y must fire in phase with the b entity node. Node Q_1 represents x and y and therefore fires in two phases, one in phase with a and one in phase with b. Q_3 represents only x and fires in phase with a, and Q_4 encodes only y and fires in phase with b. Bindings propagate to the antecedent ($P(x, y)$) layer where each node has a threshold θ of 2. P_1 represents y and so fires in phase with b, P_2 represents x and so fires in phase with a, and P_3 represents both x and y and so fires in phase with both a and b. The network is now trying to confirm $P(a, b)$?, the fact encoded for by the fact circuit. The output from P_1 is inhibited by a matching phase from b, the output from P_2 is inhibited by a matching phase from a and the two-phase output from P_3 is inhibited by matching phases from b and then a. The fact node can therefore fire uninhibited to assert that $P(a, b)$ and therefore also $Q(a, b)$ are true.

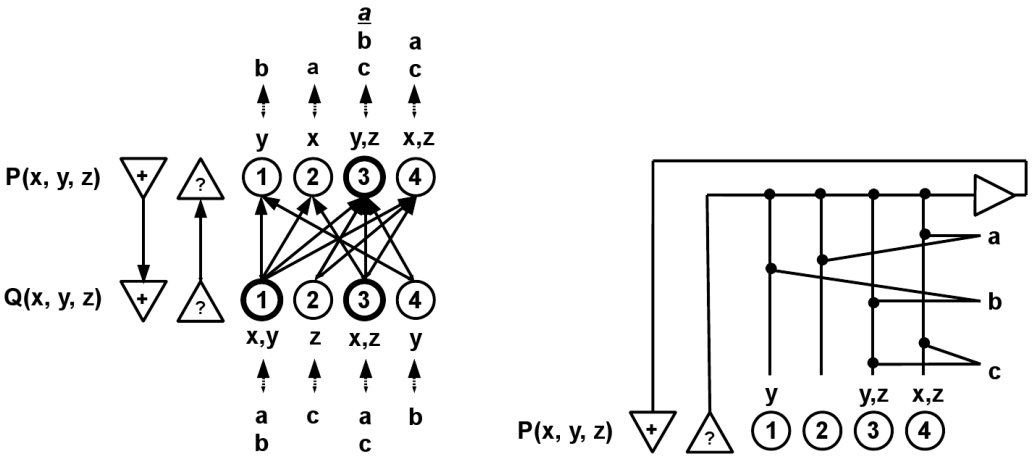
Unfortunately, one or both of two problems were encountered for some distributions of argument representations using this method. These problems identify obstacles that must be overcome in producing a distributed SHRUTI model. Either some questions were answered incorrectly or the propagation of some bindings were ambiguous so that it was not possible to tell which argument was supposed to be bound to which entity. The following subsection describes these problems and subsections 5.3.3 and 5.3.4 explore the likelihood of these problems occurring.

5.3.2. Error and ambiguity

Both problems occur in some situations when the representation of an argument r is distributed over θ nodes representing a set of arguments R for which $r \notin R$. This situation will be referred to as *θ-overlap* henceforth. For example, in figure 5.2a argument x in predicate Q undergoes θ-overlap because $\theta = 2$ and the representation of x is distributed over θ (2) nodes that represent other arguments (y and z).

Error

Although θ-overlap is not a problem in itself, it does cause variable bindings to propagate incorrectly when θ-overlap occurs in the consequent of a relation and some node in the antecedent encodes for R but not r. In this situation, θ nodes in the consequent propagate a binding of r to that antecedent node so that it is incorrectly bound to r. For example, consider the network in figure 5.2 and the question $Q(a, b, c)$? which should be confirmed as true by the antecedent fact $P(a, b, c)$ (figure 5.2b). Node P_3 encodes for y and z and therefore fires in the b and c phases, but also receives two a phases from nodes Q_1 and Q_3 which both represent x, bound to a. These inputs propagating the a phase cross the threshold $\theta = 2$ and therefore node P_3 also fires in the a phase even though it should not. The inhibitory link from P_3 to the input of the fact node is only inhibited by output from b and c, but not a. Therefore the a phase from P_3 is uninhibited and interrupts the activation of the fact circuit so that it cannot confirm that $P(a, b, c)$ is true.



(a) Relational structure for the relation $P(x, y, z) \rightarrow Q(x, y, z)$.

(b) Fact for circuit in figure to the left (P(a,b,c))

Figure 5.2. Problematic distribution. Node P_3 should only fire in phase with b and c, but also fires in phase with a because two of its inputs propagate this phase. Because this a phase is not inhibited in the fact circuit, it interrupts the activation of the fact node $P(a, b, c)$ which would activate otherwise.

	# argument nodes bound to entity				
Argument	a	b	c	Binding	Ambiguous?
x	**2 (67%)**	0 (0%)	1 (33%)	a	No
y	1 (25%)	**2 (50%)**	1 (25%)	b	No
z	**2 (40%)**	1 (20%)	**2 (40%)**	a, c	Yes

Table 5.1. The proportion of argument bindings for predicate P following propagation from Q in figure 5.2. Numbers in bold represent majority bindings for each argument. z from predicate P is bound to a as much as it is to c and therefore the binding for this argument is ambiguous.

Ambiguity

However, the problem is owed not only to the propagation of bindings but also to how the fact circuit interprets bindings. The current fact circuit considers each role node separately and inhibits the activation of the fact node when at least one role node fires in an incorrect phase. If the fact circuit were to somehow consider the proportion of bindings across all nodes representing an argument, the entity to which the majority of an argument's nodes are bound could be interpreted as the entity to which that argument is bound. It is this majority binding that could somehow be tested against the static binding in order to assert the truth of a predicate instance. Continuing the example of the circuit in figure 5.2, table 5.1 shows that for predicate P, 67% of nodes representing x are bound to a and 33% to c. x is bound to a more than it is to c and could therefore be interpreted as being bound to a. Likewise, y is bound to all values but mostly to b. However z is bound to a as much as it is to c and therefore the binding of z is ambiguous; there is no way of determining if it is supposed to be bound to a or to c. Ambiguous propagation is the second problem with this preliminary representation.

An argument binding is defined as ambiguous when two or more entities to which nodes representing that argument are bound are all bound to the maximum number of those nodes bound to any entity. For example, in table 5.1, the maximum number of nodes representing z that are bound to any entity is 2. Both a and c are bound to 2 nodes representing z, and therefore the binding is ambiguous. The total ambiguity of a network is defined as the number of predicate arguments for which this is the case.

If the distributed SHRUTI model could be modified so that argument bindings are never propagated ambiguously, and fact structures could be modified to interpret these unambiguous bindings, then the distributed SHRUTI model might be successful in answering questions without error. Although the occurrence of error and ambiguity both depend on how bindings are propagated, θ-overlap in the consequent is a necessary condition for both to occur. One might argue that with a large enough n and a small enough b, θ-overlap would be so unlikely that error and ambiguity would be equally unlikely and modifying the distributed model to overcome these problems would be unnecessary. However, with a high n and a small b the saturation of representations becomes so low that the representations rarely overlap at all and are therefore are effectively local, thus defeating the point of using distributed representations in the first place. Sections 5.3.3 and 5.3.4 present experiments performed to demonstrate this point.

5.3.3. Probability of θ-overlap: method

When a group of neurons are saturated with a large number of representations, the probability of θ-overlap ($P(\theta\text{-}overlap)$) is high. The argument put forward above is that as $P(\theta\text{-}overlap)$ is reduced to also reduce the likelihood of error and ambiguity, the model tends towards a more localist representation. Some experiments on how $P(\theta\text{-}overlap)$ changes with different values of n, b and θ were performed to demonstrate this point further, and more generally to demonstrate when error and ambiguity are likely to occur. This section presents the method used to obtain results presented in section 5.3.4. First, it is necessary to introduce some terms and explain how $P(\theta\text{-}overlap)$ is measured:

- Let r equal an argument representation of size b in a cluster of n neurons.
- R_b^n is the set of all possible r. In other words, R_b^n is the set of all possible argument representations of size b in a cluster of n neurons.
- Let c equal a set of argument representations for a predicate of three arguments so that $c = \{c_1, c_2, c_3\}$ where $c_i \in R_b^n$.
- C_b^n is the set of all possible c. In other words, C_b^n is the set of all possible predicate representations for predicates with three arguments.
- O_b^n is a subset of C_b^n containing all members of C_b^n for which θ-overlap occurs.
- $|R_b^n|$, $|C_b^n|$ and $|O_b^n|$ denote the number of elements in R_b^n, C_b^n and O_b^n.[2]
- $P(\theta\text{-}overlap) = |O_b^n|/|C_b^n|$ measures the probability that a predicate representation c contains arguments that exhibit θ-overlap.

For a range of values of n, b and θ, 10 representations (sets of argument-node assignments) were randomly generated for the relation $P(x, y, z) \rightarrow Q(x, y, z)$. Facts associated with this relation were $P(a, b, c)$ and $P(d, e, f)$ so that the conclusions of the logic program were $Q(a, b, c)$ and $Q(d, e, f)$. This small, one-relation logic program was chosen to demonstrate the simplest scenario in which error and ambiguity occur. Each network was presented with the enumerated set of all possible questions that should be answered 'true' (including $Q(a, y, z)$, $Q(a, b, z)$, etc.), and error was measured as the number of incorrect answers, the maximum of which was 14. Values for error and ambiguity for a set of n, b and θ values were averaged over the 10 different representations randomly generated for that set and then normalised. It was hypothesised that error and ambiguity would be greater for higher levels of saturation, that is for greater values of $P(\theta\text{-}overlap)$. Because both predicates include three arguments and ambiguity can only occur in the predicate to which bindings are propagated (P), the maximum possible value for ambiguity before normalisation is 3, when the bindings of all three arguments in P are ambiguous.

[2] Although $|R_b^n|$ can be computed by calculating the value of binomial coefficients, no formula for calculating $|C_b^n|$ and $|O_b^n|$ has been proposed. Instead values were calculated by brute-force by enumerating all members of R_b^n, C_b^n and O_b^n.

Experiments were also performed with and without mediator structures to explore how this affects error and ambiguity. It was expected that both would be worse when bindings were propagated via mediator structures since bindings were to be propagated twice and therefore the possibility of propagating a binding incorrectly was greater.

Experiments were also performed with $\theta = b$ and $\theta = b - 1$. $\theta = b$ does not allow for any graceful degradation since all nodes representing an argument must fire in order to break the θ threshold in the antecedent layer. $\theta = b - 1$ allows for some graceful degradation but increases the likelihood of θ-overlap since θ is smaller.

5.3.4. Probability of θ-overlap: results

Table 5.2 and figure 5.3 show how error and ambiguity vary with the probability of θ-overlap ($P(\theta\text{-}overlap)$). As expected, error and ambiguity were generally greater for a higher $P(\theta\text{-}overlap)$. Figures 5.3a and 5.3c show that on average error and ambiguity were both greater when propagating bindings through mediators because bindings are propagated twice and therefore more likely to be propagated incorrectly. Figures 5.3b and 5.3d show that values of $P(\theta\text{-}overlap)$ below 0.778 are never generated when $\theta = b - 1$

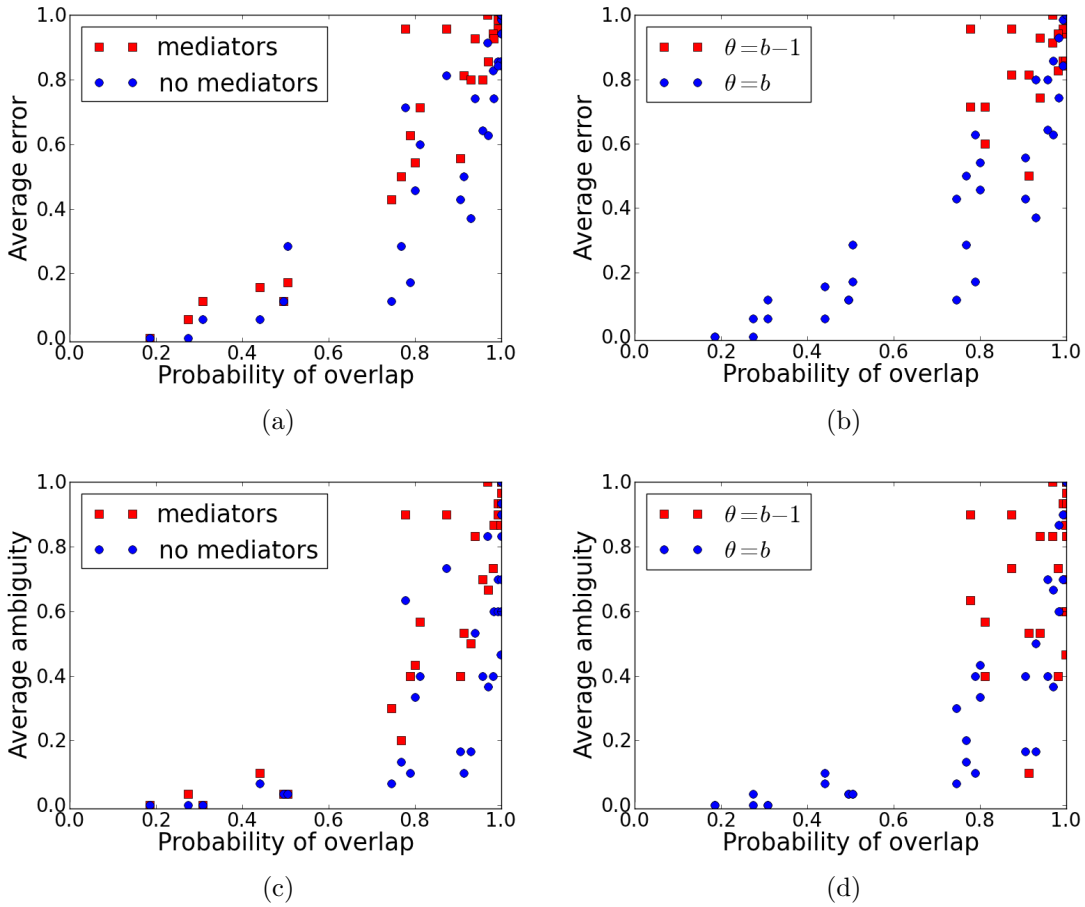


Figure 5.3. Plots of the relationship between $P(\theta\text{-}overlap)$ and average error and ambiguity. Graphs a and c demonstrate the effect of including mediator structures and graphs b and d demonstrate the effect of the relationship between activation threshold (θ) and the number of bits used to represent an argument (b).

			No Mediator		Mediator					
n	b	θ	Av. Error	Av. Ambiguity	Av. Error	Av. Ambiguity	R_b^n	C_b^n	O_b^n	P(θ-over)
4	2	2	0.46	0.33	0.54	0.43	6	120	96	0.8
	3	3	1	1	1	1	4	24	24	1
	2	1	1	1	1	1	6	4	120	1
	3	2	1	1	1	1	4	24	24	1
6	2	2	0.05	0.07	0.16	0.1	15	2730	1200	0.44
	3	3	0.17	0.1	0.62	0.4	20	6840	5400	0.789
	4	4	0.64	0.4	0.8	0.7	15	2730	2610	0.956
	5	5	1	1	1	1	6	120	120	1
	2	1	0.91	0.83	1	1	15	2730	2640	0.967
	3	2	0.94	0.7	1	1	20	6840	6840	1
	4	3	0.99	0.9	1	1	15	2730	2730	1
	5	4	1	1	1	1	6	120	120	1
8	2	2	0	0	0.06	0.03	28	19656	5376	0.274
	3	3	0.11	0.03	0.11	0.03	56	166320	82320	0.495
	4	4	0.29	0.13	0.5	0.2	70	328440	252000	0.767
	5	5	0.37	0.16	0.8	0.5	56	166320	154560	0.929
	6	6	0.74	0.6	0.93	0.86	28	19656	19320	0.983
	7	7	1	1	1	1	8	336	336	1
	2	1	0.81	0.73	0.96	0.9	28	19656	17136	0.872
	3	2	0.74	0.53	0.93	0.83	56	166320	156240	0.939
	4	3	0.86	0.6	0.96	0.93	70	328440	325920	0.992
	5	4	0.94	0.6	1	0.93	56	166320	166320	1
	6	5	1	0.93	1	1	28	19656	19656	1
	7	6	1	1	1	1	8	336	336	1
10	2	2	0	0	0	0	45	85140	15840	0.186
	3	3	0.06	0	0.11	0	120	1685040	519120	0.308
	4	4	0.29	0.03	0.17	0.03	210	9129120	4607820	0.505
	5	5	0.11	0.07	0.43	0.3	252	15813000	11781000	0.745
	6	6	0.43	0.17	0.56	0.4	210	9129120	8263500	0.905
	7	7	0.63	0.37	0.86	0.67	120	1685040	1634640	0.970
	8	8	0.84	0.7	0.99	0.9	45	85140	84420	0.992
	9	9	1	1	1	1	10	720	720	1
	2	1	0.71	0.63	0.96	0.9	45	85140	66240	0.778
	3	2	0.6	0.4	0.71	0.57	120	1685040	1365840	0.811
	4	3	0.5	0.1	0.81	0.53	210	9129120	8337420	0.913
	5	4	0.83	0.4	0.94	0.73	252	15813000	15510600	0.981
	6	5	0.85	0.47	0.99	0.87	210	9129121	9110220	0.998
	7	6	0.84	0.6	1	0.97	120	1685040	1685040	1
	8	7	1	0.83	1	1	45	85140	85140	1
	9	8	1	1	1	1	10	720	720	1

Table 5.2. Statistics recorded when generating random representation distributions for the logic program containing the relation $P(x, y, z) \rightarrow Q(x, y, z)$ and the facts $P(a, b, c)$ and $P(d, e, f)$. Error and ambiguity values are averaged over 10 different distributions for each set of values for n, b and θ, and then normalised. Error is measured as the number of questions answered incorrectly, and ambiguity is measured as the number of predicate arguments with ambiguous bindings (see page 183).

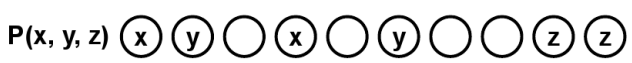


Figure 5.4. An example of a representation of P(x,y,z) distributed so widely across a set of neurons that it is effectively localist since each predicate argument is assigned to its own set of neurons. Here, $n = 10$ and $b = 2$.

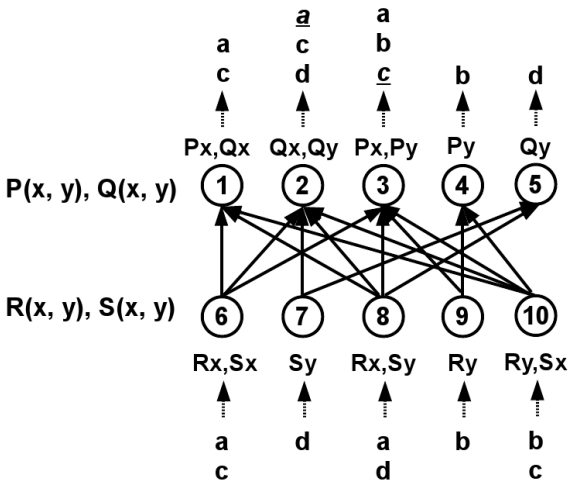


Figure 5.5. Distributing between predicates for $P(x, y) \rightarrow R(x, y)$ and $Q(x, y) \rightarrow S(x, y)$ when $\theta = 2$. When instantiating $R(a, b)$ and $S(c, d)$, node 2 incorrectly fires in the a phase, node 3 incorrectly fires in the c phase and P_x and Q_x are ambiguously bound to both a and c

because θ is lower and therefore $P(\theta\text{-}overlap)$ is higher. Error values in this case are always greater than or equal to 0.5 and with only one exception, ambiguity values are greater than or equal to 0.4. Many lower error and ambiguity values are found when $\theta = b$, suggesting that lowering the threshold relative to the number of bits used to represent an argument is problematic for propagating bindings.

The best results were obtained for $n = 10$, $b = 2$ and $\theta = 2$, in which error and ambiguity were zero with and without the use of mediators. However, in this case of particularly low saturation, it is highly likely that each argument is represented by its own set of neurons so that representations do not overlap at all. Figure 5.4 presents an example for which this is the case. This demonstrates the point made above that under the current model, it is difficult to reduce the possibility of error and ambiguous bindings without reducing the representation to one that is effectively localist. This defeats the reasons for moving towards a distributed model of SHRUTI in the first place.

These problems have only been observed for networks in which argument representations are distributed within predicates but not between them. However the same problems would still occur because θ-overlap is still likely. Consider two relations $P(x, y) \rightarrow R(x, y)$ and $Q(x, y) \rightarrow S(x, y)$ with argument representations distributed between predicates as shown in fig. 5.5. θ-overlap occurs because the representations of R_x and S_x are both distributed over θ (2) nodes representing other arguments. If the network has to simultaneously assert the truth of $R(a, b)$ and $S(c, d)$, node 2 incorrectly fires in the a phase, node 3 incorrectly fires in the c phase and both P_x and Q_x are ambiguously bound to both a and c.

Note that the way in which nodes represent roles in different predicates in fig. 5.5 is similar to the distribution of roles across nodes in CILP++ [34], described earlier in section 2.3.4.

In fig. 2.6, the representation of the role C is distributed across two nodes that each also represent different roles (A and B). It is mentioned in section 2.3.4 that although CILP++ is used to represent first-order logic, there is no neural mechanism for variable-binding. However if a solution to the problem of θ-overlap for the distributed SHRUTI model can be found so that it can propagate bindings correctly, then the resulting neurally-plausible binding mechanism could be extended to CILP++ also. Furthermore, the solution may also extend to the RTRBM [27, 28]. Like SHRUTI, the RTRBM represents probabilities in the relations it encodes, and like the hypothetical distributed SHRUTI, also takes the form of an associative memory. Integrating the distributed spiking-neuron binding mechanism with the RTRBM would extend the probabilistic reasoning capabilities of the RTRBM to first-order relations.

In summary, using the current representational method, it is difficult to reduce the probability of error and ambiguous binding propagation without reducing the level of saturation to a point where representations are effectively localist and no longer distributed. One might argue that these findings support the localist hypothesis, because moving to a distributed representation causes error in the output of the network. However, as argued earlier in section 2.1.3, even humans make errors during reasoning, and therefore an incorrect response to a question is not necessarily a reason to prefer one form of representation over another as far as cognitive modelling is concerned. It would not be fair to say that the findings support the localist hypothesis without first attempting to reduce the error yielded by distributed networks to a degree at which it is comparable to human error. The problem with the current distributed representation is that even a small degree of distribution causes error in a small network representing only one relation. This suggests that the same or even greater degree of distribution in even larger SHRUTI networks would cause similar problems for each of the relations it represents. In order to further advance the idea of a distributed network that uses spiking neurons and temporal synchrony for variable binding, measures must be take to organise the network so that bindings are not propagated ambiguously and so that fact structures (and other SHRUTI structures) can interpret and manipulate those unambiguous bindings. Solutions to these problems may also enable variable-binding in CILP++ [34] and the RTRBM [27, 28]. The compatibility of the distributed SHRUTI model with Hebbian learning will also need to be explored.

5.3.5. Organising distributed representations

One possible means of avoiding ambiguous propagation is to somehow organise the network so that this does not happen. Using the current model, argument bindings will not propagate properly under some representations but will under others. This suggests that SHRUTI supports distributed representations using associative networks as long as the representations are organised in such a way that bindings are not propagated ambiguously. This section discusses some possible directions to take in attempting to organise distributed representations.

If one were to leave the emergence of such organisation to chance, it is possible that the

parameters of the network (n, b and θ) could be adjusted so that ambiguous propagation is unlikely. However as argued in section 5.3.4, the disadvantage of this is that the representations of different arguments become so unsaturated that they may as well be localist.

Alternatively, the organisation could be enforced through learning, development, memory consolidation, or a combination of these. As a representation of a concept is learned, it and other representations would have to be organised so that bindings are not propagated ambiguously. Alternatively, when structures become too saturated or knowledge becomes ambiguous, existing representations could be restructured.

In many distributed representations, the meaning associated with a neuron is defined by the connectivity it shares with other neurons. Using this idea, excitatory or inhibitory connections between neurons in the same layer may help to consolidate variable bindings. For example, a neuron A representing both P_x and Q_y could be inhibited by another neuron B that only represents P_x so that in this particular instance, A only fires as P_x. An input layer would be presented with a set of bindings, these would be propagated to the middle layer (mediators), in which bindings are consolidated through the interconnections between nodes, and then propagated to the output layer, in which bindings are once again consolidated by the same means.

5.3.6. Other nodes

The distributed representations discussed so far only concern distribution among role nodes. SHRUTI relations also involve propagation between enablers and collectors. However because these do not require bindings to be performed, the distribution of information across enablers and collectors using associative memories is equivalent to the distribution of representations of propositional logic programs over associative memories, for which solutions have already been proposed [3, 109].

However the difficulty with evolving SHRUTI mediator structures in the chapter 4 was that different components of the relational structure were required to be discovered together in order to influence objective fitness. Even if a working associative network for propagating role bindings did exist, it would be unable to influence fitness until associative memories for enablers and collectors had also been discovered.

In other words, each discovery must make its own contribution to objective fitness. As there are three different functions to consider in SHRUTI relational structures (enabler, collector, and role node connections), these three functions may also have to be distributed across the set of connections in addition to distributing bindings and instead of assigning each function its own associative memory. In order to do this, the same would need to be true of the nodes at each layer. Rather than performing only one function, a node would provide some contribution to each function. For example, a node may be primarily a role node but also provide some contribution to the activation of enablers. Discovering connections between two role nodes in the original SHRUTI model would

provide no contribution to objective fitness without enabler and collector connections already being established. However, in the hypothetical distributed model, discovering connections between two nodes that were primarily role nodes but also participated in the propagation of collector and enabler connections would have some influence on fitness.

5.3.7. Other SHRUTI structures

The ideas discussed in this chapter so far only concern SHRUTI's relational structure. Evolutionary experiments in chapter 4 showed that fact structures were also difficult to evolve for essentially the same reason; owing to the discrete nature of the fitness landscape, it contains insufficient information to influence an evolutionary search. Because SHRUTI's structures in general use localist representations, it is expected that similar problems may be encountered attempting to evolve any of SHRUTI's structures.

The fundamental contribution of SHRUTI to neural-symbolic reasoning is the use of spiking neurons and temporal synchrony to propagate dynamic variable bindings as performed by SHRUTI's relational structures. Once a distributed equivalent to this mechanism has been produced that enables unambiguous propagation of variable bindings, then distributed equivalents of fact structures and other SHRUTI mechanisms that are compatible with the new relational structure can be considered and tested for evolvability.

5.3.8. Evolvability

Distributing SHRUTI representations in the ways discussed above does not necessarily guarantee an improvement to the evolvability of SHRUTI. The argument is rather that this is a worthwhile solution to explore considering hypotheses as to why evolution struggled in chapter 4. Each of the changes proposed above works towards a model in which each neuron encodes for multiple concepts and functions, and each concept and function is encoded by multiple neurons, so that each structural discovery provides its own contribution to objective fitness. The resulting fitness landscape, now more continuous or at least more granular than before, would contain more information that may be useful in helping an evolutionary search discover fitter genomes, and this is why distributed representations are a logical step to take in attempting to improve the evolvability of SHRUTI networks.

A means of testing improved evolvability will of course be required. Genomes would be evolved as in chapters 3 and 4, by running the NSGA algorithm with the objectives set to minimise e-area and the number of weight updates. However, measuring error may be difficult in the distributed context. Whereas in chapters 3 and 4 error could be determined by measuring the output of a predicate's collector nodes, in a distributed model multiple nodes may participate in representing the answer for a given predicate. Exactly which nodes do this may be different for each network, depending on how it develops, which itself depends on the instructions encoded by the evolved genome. A means of locating nodes that participate in the distributed representation of output corresponding to a particular predicate will be required before an answer to a question can be determined. Zero-error

genomes can be sampled as described in section 3.3.6 in order to explore their genotypic and phenotypic structure. Even if zero-error networks are still not discovered by searching the space of distributed networks, this does not necessarily mean that evolvability has not improved. Larger hypervolumes than those yielded by the searches in sections 4.3.5 and 4.4.4 would indicate improved evolvability, because this would indicate that the Pareto fronts have progressed closer to the minimum objective values than they did in the search for localist SHRUTI networks.

Although an early distributed alternative to the SHRUTI model has been presented in section 5.3.1, it was not worth attempting to evolve this model using the method described above because the model is highly prone to error (section 5.3.4), owing to the θ-overlap problem. As soon as a solution to this problem has been found such that inference can be performed with reasonably low error, then it would be worth investigating the evolvability of the model.

5.4. Visuospatial and linguistic systems

This section considers a possible alternative approach to discovering distributed, biologically plausible models of reasoning that would also support existing neuroscientific theories of reasoning. Findings in the neuroscience literature suggest that reasoning may take place in regions of the brain largely associated with visuospatial information or linguistic and syntactic processing [35, 81]. Therefore an interesting research direction to take would be to see if distributed neural models for natural language processing [15, 70, 101] and visual attention [10, 50, 74] can be adapted to learn facts and relations and answer 'true or false' questions presented in the form of predicates and argument bindings, just as SHRUTI does. The fact that associative memories have been applied to both spatial manipulation [121] and reasoning [3, 109] provides some support to this idea. Furthermore, successful adaptation of visuospatial or linguistic models in this way would support the hypothesis made in the neuroscientific literature about deductive reasoning being performed in the brain regions involved in visuospatial and linguistic processes.

5.5. Summary

Distributed alternatives to SHRUTI networks are worth exploring because they may improve the evolvability of SHRUTI networks and at the very least improve the biological plausibility if, as evidence suggests, the human brain uses distributed [80] or partially distributed [77] representations. The hypothetical distributed SHRUTI representation will allow individual structural discoveries to make their own contribution to objective fitness, thus adding more information to the fitness landscape and also smoothing the transition between developmental stages. The information processing theory of cognitive development [69] is compatible with distributed representations and is among theories of development that support more continuous transitions between developmental stages [100].

Successfully evolving SHRUTI networks would in itself contribute towards biological plausibility by demonstrating that SHRUTI networks can be produced through the biologically plausible means of evolution and development. Of course, until this has proven successful, there is no guarantee that distributed representations will indeed improve the evolvability of SHRUTI. Rather, the aim of future work is to answer the following questions:

1. Given that there is no continuous transition between stages of SHRUTI development, can development of SHRUTI and other neural-symbolic reasoning models be made more continuous by making the structures more distributed?

2. If so, does this make them more evolvable?

This chapter outlined what problems must be overcome in order to answer these questions, with some preliminary results highlighting particular problems encountered when propagating variable bindings in associative memories. It is proposed that future work be carried out in the following stages:

1. **Associative memories**:[3] Of the existing neural-symbolic models of reasoning, associative memories are a suitable method to explore with respect to distributing SHRUTI relations since they provide a means of distributing the representation of logical relations.

2. **Spiking associative memories:** The next step is therefore to adapt associative memories to employ SHRUTI's variable binding method. However the findings in section 5.3.4 present a challenge in that it is difficult to do this without propagating variable bindings unambiguously. A solution to this problem must be found, perhaps by somehow enforcing the organisation of representations.

3. **Spiking associative SHRUTI relations:** Next, this idea should be implemented into SHRUTI and a means of distributing the function of nodes in addition to the concepts they represent should be found so that each discovery has the potential to provide its own unique contribution to objective fitness.

4. **Distributed SHRUTI networks:** Finally, the distributed relational mechanism should be combined with other SHRUTI functionality such as fact structures and type hierarchies, which will need to be modified so that they can interpret the distributed bindings propagated unambiguously through the relational structures.

Finally, the hypothesis that distributed representations will improve the evolvability of SHRUTI and neural-symbolic networks in general needs to be tested. The evolvability of each of the above stages should be tested separately in the order listed. Unsuccessful evolution of any one of these stages would suggest that distribution is not the solution to evolvability after all and that attempts to evolve the following stages are unnecessary to conclude this. Nonetheless, successful distribution would still support the biological plausibility of SHRUTI networks by demonstrating that variable binding by temporal synchrony is compatible with both localist and distributed theories of representation. Furthermore,

[3] Of course, associative neural-symbolic reasoners already exist and do not need to be reinvented, but are listed as a step in this overall process for the sake of completeness

if the problems caused by θ-overlap are solved, the variable-binding mechanism used in the hypothetical distributed SHRUTI model may also extend to CILP++ [34] and the RTRBM [27, 28], owing to structural and representational similarities these models share with the distributed SHRUTI. This would enable variable-binding to be performed in both models, and in the latter case, would enable neural-symbolic probabilistic reasoning to be extended to first-order relations.

The theory that deductive reasoning is performed in visuospatial and linguistic regions of the brain suggests another possible direction for future research. The possibility of adapting models of these regions to perform reasoning could be explored, as could the possibility of evolving these adaptations. Doing so would not only contribute towards a biologically plausibly model of reasoning but also support neuroscientific theory.

6. Conclusions

This chapter concludes the thesis by reviewing the overall findings with respect to the aims of this research (section 6.1), highlighting the contributions they make to neural-symbolic integration and generative and developmental systems (section 6.2), and discussing future work (section 6.3). For clarity, the aims and claims of this thesis as listed in chapter 1 are repeated below:

Aims:

1. To test the claim of SHRUTI's developers that owing to SHRUTI's use of repeated, similar sub-circuits, it is suited to representation by indirect encoding that enables the circuits to be pre-organised in such a way that enables its structures to be learned.

2. To argue that by using such a biologically plausible representation, a stronger case can be made for the biological plausibility of the SHRUTI model.

3. To investigate the biological plausibility of the SHRUTI model further by exploring the evolvability of SHRUTI networks under the proposed genome representation.

4. To argue a case for exploring the possibility of improving the evolvability and therefore also the biological plausibility of SHRUTI networks by moving from a localist representation to a more distributed one.

Claims:

1. SHRUTI structures can be represented by indirect encodings and therefore the biological plausibility of SHRUTI networks is supported by an encoding method akin to that used by DNA.

2. Simple SHRUTI structures can be discovered through an evolutionary process and the discovered genomes that develop those structures are both scalable and adaptable to logic programs other than those they were trained on.

3. The discovery of more complex SHRUTI structures through evolution is highly unlikely because the fitness landscape lacks the information necessary for identifying individual components that improve fitness when combined in later generations.

4. Because the difficulty in identifying these components is likely to be a consequence of the localist encoding used by SHRUTI, a move to distributed representation is a worthwhile avenue of exploration in attempting to improve the evolvability.

6.1. Overview of findings

Sections 3.2, 4.3.3 and 4.4.3 report that the representation of SHRUTI relational and fact structures in a biologically plausible genome was successful, thus meeting the first two aims and supporting the first claim. The SHRUTI genome exhibits biological plausibility in a number of ways. Most significant is the use of indirect encoding itself and the reproduction of repeated, similar structures that allows for scalability. DNA behaves in a similar way, providing genetic instructions for the gradual development of an organism [22, 103, 116]. Also like DNA, the SHRUTI genomes employ gene regulation in which the expression of one gene results in the expression of another [116], although some genes known as *introns* may not be expressed at all. Other biologically plausible facets of the model include neurogenesis [1, 12, 123], activity-dependent development [33] and the diffusion of chemicals in such a way that chemical levels can be interpreted as positional information that influence development (cell differentiation [116]). However, the genome is not completely biologically plausible as it is still an abstraction and does not match the complexity of real biological genomes. Particular areas for improvement with respect to biological plausibility are discussed in the future directions section below (section 6.3).

Chapters 3 and 4 also explored the evolvability of SHRUTI networks. This was the third aim of the thesis. The findings in section 3.4 show that the evolution of SHRUTI genomes was successful for a simple relational structure in which no mediators are included because the genome itself was rather simple and required a minimum of one condition and action to produce zero-error networks that could answer all test questions correctly. Furthermore, genomes evolved for one logic program adapted well to others, owing to the very nature of indirect encodings. The genomes were not evolved to represent any particular network structure, but were evolved to represent a repeatable relational structure that was transferable to any logic program. This is what enabled the evolved genomes to be so scalable and adaptable, and the results demonstrate the suitability of indirect encodings for the representation of logic programs and repeated substructures in general. These findings support the second claim of this thesis.

Although evolution could not discover SHRUTI's mediator and fact structures, as shown in sections 4.3.5 and 4.4.4, examination of the target network structures and the fitness landscape in sections 4.3.6 and 4.4.5 suggested that the brittle nature of SHRUTI representations was responsible, and that distributed encodings would be a worthwhile avenue to explore in future work. Whereas the non-conjunctive relational structures required as few as one condition and action in the genome, mediator and fact structures both required many more conditions and actions to produce a set of necessary substructures. According to the building block hypothesis, complex structures emerge through evolution by the discovery of necessary components or 'building blocks' in earlier generations [36]. However, these components must be recognisable in the fitness landscape. Reproduction of particular structures in the field of generative and developmental systems is known to be difficult because the fitness landscape is deceptive in that discovery of these individual components in genotype space is not often reflected by a positive change in objective fitness [122]. By recording the objective values yielded by controlled and random mutations of SHRUTI

mediator and fact genomes, it became clear that this was also the case when attempting to evolve indirect encodings of SHRUTI. Furthermore, the fitness landscape is heavily populated by group 3 genomes that always answer 'unknown' to any questions they are asked, and even if these are close to working SHRUTI genomes in genotype space, such proximity is not reflected in objective fitness space. In summary, an evolutionary search will struggle to discover particular SHRUTI structures because the fitness landscape contains insufficient information to identify the individual discovery of necessary components or proximity to those components. This supports the third claim made in this thesis.

Chapter 5 discussed a possible means of improving evolvability and therefore concerns the fourth aim of this thesis. The discovery of mediator and fact components is difficult because they must be discovered together and not individually in order to influence objective fitness. For example, in the mediator evolution, enabler-to-enabler or collector-to-collector mediators must both be discovered in order for activation to propagate from a predicate's enabler, through the network of relations and facts, and back to that predicate's collector. If this is not possible, the collector will not fire and the network will always answer 'unknown'. Owing to the localist representations in SHRUTI, the mapping of a function to a node is one-to-one (or one-to-many in the case of ensembles). Collector nodes only encode the collector function, enabler nodes only encode the enabler function, and neither can influence fitness alone. If the network is designed so that each discovery in genotype space could make some individual contribution to fitness, the landscape may be informative enough to direct the search more effectively. In a distributed encoding, a relational structure or a component of that structure (role, enabler, collector) would not be encoded explicitly by a particular neural structure. Instead, any neuron could encode for multiple concepts or node functions so that the discovery of each neuron would be more likely to provide its own contribution to fitness. Therefore the fourth claim of this thesis is that exploration of distributed encodings would be worth investigating as a possible means of improving the evolvability and therefore also the biological plausibility of SHRUTI networks. Note that localist representations may be partially distributed, and therefore the argument here is not that SHRUTI may need to be fully distributed, but that it may need to at least be *more* distributed. Chapter 5 also presented preliminary results that highlight problems that must be overcome when distributing SHRUTI representations, but these are reviewed in section 6.3.

Note also that this thesis does not claim that a move to distributed representations will definitely improve evolvability, only that the findings suggest that this is the next logical step to take with respect to exploring evolvable SHRUTI networks. However even if evolvability is not improved, there are additional advantages to producing a distributed model of SHRUTI. In particular, a stronger case for biological plausibility could be made by demonstrating that SHRUTI's spiking temporal binding mechanism is compatible with both localist and distributed theories of neural representation. Furthermore, in sections 3.2.4 and 4.3.3 it was argued that it is not possible for a SHRUTI genome to restrict the size of the network without restricting the development of a relational structure to a time when evidence for it is observed. This makes the learning algorithm redundant and makes development fully dependent on the evidence on which learning operates. A

distributed encoding may enable a different developmental paradigm similar to the information processing theory of development [69, 100]. Development could instead serve to meet particular storage capacity or processing demands rather than developing particular structures to reflect particular observations. For example, when the network becomes saturated with a large number of distributed representations, development could produce more neurons and connections over which to distribute new concepts. Furthermore, the information processing theory is one of many theories that supports a more continuous transition between developmental stages [100], as opposed to Piaget's suggestion of more discrete stages [78]. The stages in which SHRUTI develops under the current localist representation are also discrete, and a move to a distributed encoding based on the information processing theory would be more fitting with modern theories of continuous transition between stages.

6.2. Contributions

This section highlights the contributions that the above findings make with respect to the motivations of this work and the fields of neural-symbolic integration and generative and developmental systems.

There were three motivations to this work. The first and primary motivation was that SHRUTI's developers claim that because biological genomes contain indirect encodings of repeated substructures, SHRUTI networks, which are also composed of repeated substructures, can also be represented by indirect encodings and therefore such a representation would complement the biological plausibility of the SHRUTI model [89, 95]. The primary aim of this thesis was to support the biological plausibility of SHRUTI in this respect by making the first attempt at representing SHRUTI in this way and was successful in doing so. However to extend the biological plausibility further, a biologically plausible genome should also be evolvable. The evolvability of the SHRUTI genome was found to be limited in that only simple structures could be produced through evolution and more complex structures could not be discovered because they require multiple components to be discovered at the same time before they can influence the performance of the overall network. This appears to be a consequence of the localist representation used to represent them and therefore distributed representations are a worthwhile avenue to explore in attempting to improve the evolvability and therefore also the biological plausibility of SHRUTI networks.

The second motivation was that one of the goals of producing neural-symbolic reasoning models in general is to discover neural mechanisms that enable reasoning. Neuroscience is currently developing an understanding of how reasoning is performed at the higher anatomical level [81], but cannot yet isolate particular regions of the brain so that an understanding of how the neurons themselves enable reasoning can be achieved. Until then, neural-symbolic models of reasoning can only suggest how this may be possible. One means of discovering such models may be to discover them through evolution, but before this idea is considered further it is necessary to explore how feasible it is. This

thesis begins this process by exploring the evolvability of SHRUTI, chosen because it is an existing neural-symbolic model which exhibits some degree of neural plausibility (biological plausibility at the neural level) by virtue of the fact that it uses spiking neurons [64] and Hebbian learning [41]. Because the localist nature of SHRUTI networks makes it difficult to evolve them, and because other neural-symbolic reasoners adhere to the localist general relational structure (section 2.3.3), this suggests that if neurally plausible models are to be explored though evolution, a search in the space of fully localist models may be unfruitful and that the space of distributed representations should also be explored. This is not to say that localist representations should be completely ruled out. Some evidence for localist representations in the brain still exists and even proponents of the localist hypothesis argue that localist models may be partially distributed [11, 77]. Partial distribution may be sufficient to improve evolvability, and future experiments should explore a space of networks in which networks have the freedom to be localist or distributed to varying degrees. Furthermore, difficulties encountered in evolving fully localist SHRUTI networks that exhibit no distribution at all suggest that natural evolution may also struggle to discover localist models, thus supporting arguments for some degree of distributed representation in biological neural networks.

The third and final motivation, related to the previous, is that neuroevolution in general has been suggested as a means of discovering neural models of general cognition [55, 71, 85, 106]. Furthermore, it has been suggested that this may be possible in a biologically plausible way by representing networks as generative and developmental systems, which like DNA, define a set of genetic instructions for the gradual development of the phenotype instead of describing it explicitly (direct encoding). Although currently concerned with development of solutions to control problems, the evolution of generative and developmental systems may one day produce general models of intelligence. Among the abilities that such models should exhibit is the ability to reason. Furthermore, because of the similarities that reasoning shares with other cognitive processes, results of attempting to evolve reasoning systems may be indicative of the evolvability of cognitive structures for performing other tasks. For example, findings in neuroscience suggest that reasoning ability is related to visual and linguistic ability [81]. Furthermore, parallels between the learning of relations and the learning of associations between visual scenes were drawn at the end of section 2.3.6 on page 56. As the previous paragraph has already argued, the emergence of neural reasoning or analogous processes in a general model of intelligence may be unlikely if limited to fully localist representations only and therefore these general models should be evolved with the capacity for distributed representations if they are to be explored properly.

6.3. Future work

Future work is divided into four categories. Firstly, the next major step of this research is the exploration of a distributed alternative to SHRUTI, so this is discussed first in section 6.3.1. There may also be improvements that can be made to the existing localist model

in order to improve its functionality and gain an even deeper understanding of the results obtained from attempting to evolve it. These are described in section 6.3.2. Improvements that could be made to the biological plausibility of SHRUTI networks and genomes are discussed in section 6.3.3, and section 6.3.4 explains how the findings of this thesis suggest new avenues of exploration in the wider field of neural-symbolic integration as a whole.

6.3.1. Distributed SHRUTI model

The next key step to take in this research is to explore whether or not the SHRUTI model is compatible with a distributed representation. The best approach to take would be to integrate the SHRUTI model with associative memories [108], as other existing distributed neural-symbolic reasoning models only distribute concepts and not the mechanisms which reason upon those concepts. Problems that must be overcome, as highlighted in section 5.3, include the issue of how to propagate variable-bindings unambiguously and how to distribute node function to make transition between the discovery of different SHRUTI substructures more continuous and therefore less discrete. These associative SHRUTI networks should then be integrated with distributed equivalents of fact structures and other SHRUTI structures not covered in this thesis [89, 95].

It is explained in section 2.1.3 that human reasoning ability is not perfect, and therefore even though the measures listed above attempt to reduce the error in a distributed SHRUTI network, some error is to be expected when attempting to produce a biologically plausible model of human cognition. It would be interesting to compare the error yielded by a distributed SHRUTI network against errors made by humans in similar reasoning tasks. Furthermore, the level of human error could be used as a benchmark when developing these networks.

Of course, successful implementation of a distributed SHRUTI model does not guarantee that the model will be more evolvable; this thesis only argues that distributed representations are a worthwhile avenue of exploration in attempting to improve evolvability. Even if zero-error genomes are not produced, improved evolvability would be identifiable by the emergence of Pareto fronts with larger hypervolumes than were yielded by the experiments in chapter 4. It was not worth attempting to evolve the distributed model presented in section 5.3.1 because it was only a preliminary model created to explore what problems are encountered when attempting to distribute representations in SHRUTI networks, and the model was very prone to error. Once the problems encountered have been overcome as described above, then the evolvability of the new model can be explored by the means suggested in section 5.3.8, in that genomes for distributed SHRUTI networks can be evolved using the same method as was used in chapters 3 and 4. However, a means of locating the output nodes that participate in providing the response to a particular question must first be found, so that the error and therefore the objective fitness of evolved networks can be measured. One of the goals of evolving SHRUTI networks was to discover alternatives to the SHRUTI model, but the evolutionary searches conducted in chapters 3 and 4 did not yield any such networks. A search of the space of distributed networks may be more

successful in this respect, and therefore the genotypic and phenotypic structures of the zero-error genomes should be investigated to determine whether or not this is the case. Any zero-error genomes discovered can be sampled by the means described in section 3.3.6.

6.3.2. Localist SHRUTI model

Although the next main step of this research is a move to a distributed model, there are still improvements that could be made to the localist model and the experiments performed on them, in order to improve the functionality of the localist model and also to better understand the results obtained in chapters 3 and 4.

With respect to functionality, although the learning algorithms used in this research were sufficient to allow target facts and relations to be learned in structures produced by development, there are still a number of improvements that could be made. In particular, the causal Hebbian learning algorithm lacks a means of learning conflicting relations in noisy event sequences (section 3.1.2), the learning algorithm for mediator structures strengthens a few redundant connections (section 4.3.1), and the fact learning algorithm lacks a means of 'forgetting' facts (section 4.4.1). Furthermore, if a more distributed alternative to the SHRUTI model is successful, the Hebbian and recruitment learning algorithms will also have to be made compatible.

Experiments were restricted to logic programs containing predicates with no more than three arguments, in order to limit the complexity of evaluating evolved networks.[1] This enables three variable bindings to be performed simultaneously, whereas humans are believed to be capable of performing approximately seven at any one time [95]. Section 3.5 demonstrates that the SHRUTI genome is capable of developing relational structures for relations containing predicates with up to seven arguments, owing to the fact that indirect encoding enables a structure (in this case a connection between two role nodes) to be encoded once but expressed multiple times. This has yet to be tested on larger logic programs containing predicates with as many arguments, but it is hypothesised that because the genome is scalable and can develop a relational structure for one seven-argument relation, it would successfully do the same for all such relations. Most evolved genomes were also shown to be adaptable to the seven-argument relation, but evolution of genomes using logic programs with such large predicates as training data was not attempted because of the computational expense of doing so. This should be attempted in future work in order to demonstrate that SHRUTI networks capable of performing as many variable bindings as humans can also be discovered through evolution.

Although genomes evolved on fixed event sequences for non-conjunctive relations generalised reasonably well to probabilistic event sequences in section 3.4.3, they did not generalise quite as well as expected and generalisability could have been improved by evolving the genomes on probabilistic sequences in the first place. However this would increase network evaluation time. Because of the probabilistic nature of the training sequences,

[1] See page 71 for a defence of this decision.

it would be necessary to train each genome on multiple sequences in order to produce a reasonable assessment of the network it develops.

Finally, although an uninformative fitness landscape was shown to be responsible for the difficulties faced by an objective-based search for complex SHRUTI genomes in chapter 4, *novelty search* [61] is proposed to be a solution to the problem of deceptive landscapes in generative and development systems. Instead of rewarding solutions that satisfy certain objectives, novelty search rewards solutions that are unique when compared to existing members of the population, and has been shown to discover fit solutions anyway because those fit solutions are also novel. A search for SHRUTI genomes using novelty search would be worthwhile. However, the biological plausibility of such a search is questionable, as it seems unlikely that organisms in nature survive simply by being unique.

6.3.3. Biological plausibility

Even though the biological plausibility of the SHRUTI model has been improved by producing a genome for developing SHRUTI networks that is itself based on biologically plausible principles, the biological plausibility of the genome is limited. In particular, the genome is still an abstraction of the much more complex biological genome. Biological genomes employ gene regulation in a network, whereas the SHRUTI genome does so in a tree. Although biological neurogenesis involves activity-dependent development [33], it is unlikely to be as direct as it is in the SHRUTI genome, which produces connections as soon as a neuron is activated. Because of the localist nature of the SHRUTI model, it is impossible to guarantee the existence of a relational structure without predisposing the genome to construct it or only constructing it when evidence for it is presented (sections 3.2.4 and 4.3.3). Both possibilities lack biological plausibility, the former because it suggests that all possible knowledge is innate and the latter because it makes learning redundant. Furthermore, the latter approach means that development is dependent on learning, whereas a significant amount of neurogenesis occurs in the pre-natal stage, before learning can even take place [1, 12, 123]. In summary, although SHRUTI networks have been shown to be representable in a biological genome as its developers suggest, work remains to be done to improve the biological plausibility of this model even further. The use of such an abstract genetic model was chosen because SHRUTI itself is also an abstraction, and therefore less abstract genetic models could be explored as the biological plausibility of SHRUTI networks is also improved.

6.3.4. The goals of neural-symbolic integration

The three goals of neural-symbolic integration listed on page 34 are the discovery of new biologically plausible models of symbolic representation in the brain, the production of mechanisms that combine learning and reasoning to solve real-world problems, and to develop methods of knowledge extraction. This subsection explains how the findings of this thesis provide new avenues of exploration with respect to these goals.

CILP [16] and the RTRBM [27, 28] already make strong contributions to the second of these goals in that they provide solutions to a range of real-world problems [8, 27, 28, 59]. CILP has recently been extended to first-order logic in the form of CILP++ [34], and the RTRBM provides a mechanism for probabilistic reasoning, yet neither method offers a neural mechanism for variable-binding. Section 5.3 presents an early attempt at combining variable-binding, spiking neurons and distributed representations. CILP++ distributes representations of predicate arguments across its nodes in a similar way to the distributed SHRUTI model, and RTRBM is a form of associative memory, the model of distribution on which the distributed SHRUTI model is based. Therefore if a solution to the problem of θ-overlap is found, the use of spiking neurons to perform variable-binding in the distributed SHRUTI model may extend to CILP++ and RTRBM as a mechanism for variable-binding in these models also. Because of the conflicting relations problem (section 3.1.2), SHRUTI struggles as a probabilistic reasoner. However, an RTRBM capable of performing variable-binding may be a more promising neural-symbolic model of probabilistic reasoning on first-order logic.

If the distribution of representations in SHRUTI networks successfully improves evolvability, then the next step is to extend these ideas to neural-symbolic reasoning systems in general. This may contribute to either of the first two goals listed on page 34, by enabling the discovery of new biologically plausible models of symbolic representation, or by enabling the discovery of new neural-symbolic solutions to real-world problems. Because SHRUTI's relational structures adhere to the general relational structure shared by other neural-symbolic reasoning models (section 2.3.3), similar difficulties that were encountered evolving SHRUTI may be encountered in any attempt to evolve these also. However, by the same reasoning, if the distribution of representation improves the evolvability of SHRUTI then it may also extend to neural-symbolic reasoners in general. SHRUTI and connectionist reasoners make different contributions to neural-symbolic reasoning. The former adds a biologically plausible binding mechanism to the general relational structure and the latter shows how the general relational structure can be extended to a range of human reasoning abilities that could interact through the fibring of networks. Demonstrating the evolvability of all neural-symbolic reasoners would contribute to an overall understanding of how general reasoning ability can emerge through evolution. Of course, the argument that problems and solutions regarding the distribution and evolvability of SHRUTI networks extend to general neural-symbolic models is not conclusive but only speculation based on current findings. What this thesis does conclude is that SHRUTI cannot be evolved under its current representation and the reasons for this should be taken into account in future attempts to improve its evolvabilty or evolve other neural-symbolic models.

Whether the goal of evolving neural-symbolic reasoning systems is to produce models of cognition or to discover models that can be applied to real-world problems, a means of determining whether or not something new has been evolved will be required. As with SHRUTI networks, it will be necessary to sample evolved networks and examine their genotypic and phenotypic structures. The method used for sampling evolved SHRUTI networks can perhaps be used (section 3.3.6), but once again a means of locating the outputs of evolved networks in order to determine fitness will be required. Knowledge

extraction methods [51] may be useful for understanding the structure of new networks that are discovered.

Turning this last point around, it may also be possible to use evolution as a means of knowledge extraction, and this idea should be explored further. Results in section 3.4.2 show that networks with the minimal number of connections required to represent a logic program can be evolved by setting the minimisation of the number of connections as an evolutionary objective. It is argued on page 105 that this approach could be used to evolve a set of simplified alternatives to a network N that minimise the number of connections required to perform the same function as N. The topology of these simplified alternatives would be analogous to a description of the fundamental logic of N's behaviour. Therefore interpreting the behaviour of these simplified networks and extracting knowledge from them would be an easier task than doing so in N.

Finally, concerning the first goal once again, neuroscientific evidence suggests that reasoning is performed in regions of the brain concerned with visual and linguistic processes [81]. To support this theory and to explore other possible neural-level representations of reasoning, this theory could be tested in a computational model by adapting any existing models of visual or linguistic processes to represent reasoning. The evolvability of such models could also be explored.

Appendices

A. Question sets and event sequences

A developed network is tested by learning logical relations based on observation of an event sequence and then answering a set of questions on the target relations. Questions and event sequences used for testing networks for each logic program are presented here.

For Neg1-4 and NoNeg1-4, two sets of questions are defined: an A set and a B set. These are used to test the development of relational structures in chapter 3. NoNeg5-8 from chapter 4 also have A and B questions sets, which are used to test the development of mediator relations (section 4.3), but also have a C question set for testing the development of fact structures (section 4.4). Networks may also be tested on an enumerated set of all possible questions that can be asked of each predicate. Unlike the predefined sets presented below, these enumerated sets are generated by brute force and are only used to test how well evolved networks generalise to questions they were not trained on.

The A set contains one question on the consequent of every relation in the logic program and is used to measure the performance of evolved networks, as it is shown in section 3.4.3 that one question per relation was the minimum required to assess the efficacy of an evolved network. The B set contains every question expected to be answered 'true' or 'false' for all relations and is used to demonstrate the efficacy of the learning algorithm in sections 3.1.1, 3.1.2 and 4.3.3. The C set contains every question expected to be answered 'true' for each fact to demonstrate the efficacy of the learning algorithm and to measure the performance of evolved fact structures. All sets contain one question expected to be answered 'unknown' for each predicate in order to demonstrate that networks do not yield corrects answer by simply always answering 'true' or 'false' for any given predicate, and to discourage the evolutionary search from discovering networks that behave in this way.

Fixed event sequences present series of observations that are repeated multiple times to support target relations to be learned by the learning algorithm. Probabilistic event sequences perform the same purpose, but observations are generated at random based on prior and relational probabilities. The prior provides the probability of an observation occurring regardless of whether that observation is the consequent of a relation, and the relational probability specifies the probability that the consequent will be observed after the antecedent is observed. For both sequences, an interval of $tW-1$ (where tW represents the window of synchrony) is added between each observation. For example, although the fixed sequence for NoNeg1 is 9 observations long, the actual sequence for $tW = 2$ will be 18 observations, with one extra interval added for each observation. Note that any time step that does not include a predicate observation is nonetheless regarded as an observation, albeit an empty one.

A.1. NoNeg data sets

A.1.1. NoNeg1

True	Unknown
$Q(a,b)$	$P(e,f)$
$R(e,f)$	$Q(b,a)$
	$R(b,a)$

(a) A set (5 questions)

True		Unknown
$Q(a,b)$	$R(c,d)$	$P(e,f)$
$Q(c,d)$	$R(e,f)$	$Q(b,a)$
$R(a,b)$		$R(b,a)$

(b) B set (8 questions)

Figure A.1. Question sets for NoNeg1

1. P(x,y)	6. R(x,y)
2. Q(x,y)	7. ...
3. R(x,y)	8. R(x,y)
4. ...	9. ...
5. Q(x,y)	

Figure A.2. Fixed event sequence for NoNeg1

Predicate	Prior
$P(x,y)$	0.2
$Q(x,y)$	0.2
$R(x,y)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x,y) \rightarrow Q(x,y)$	0.9
$Q(x,y) \rightarrow R(x,y)$	0.75

(b) Relation probabilities

Figure A.3. Probabilistic event sequence for NoNeg1

A.1.2. NoNeg2

True	Unknown
$Q(a,b)$	$P(c,b,a)$
$R(b,c)$	$Q(f,g)$
$S(a,b)$	$R(d,e)$
	$S(f,g)$

(a) A set (7 questions)

True	Unknown
$Q(a,b)$	$P(c,b,a)$
$R(b,c)$	$Q(f,g)$
$S(a,b)$	$R(d,e)$
$S(d,e)$	$S(f,g)$

(b) B set (8 questions)

Figure A.4. Question sets for NoNeg2

1. P(x,y,z)	5. Q(x,y)	9. ...
2. Q(x,y), R(y,z)	6. S(x,y)	10. R(y,z)
3. S(x,y)	7. ...	11. ...
4. ...	8. S(x,y)	

Figure A.5. Fixed event sequence for NoNeg2

Predicate	Prior
$P(x,y,z)$	0.2
$Q(x,y)$	0.2
$R(y,z)$	0.5
$S(x,y)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x,y,z) \rightarrow Q(x,y)$	0.75
$P(x,y,z) \rightarrow R(y,z)$	0.9
$Q(x,y) \rightarrow S(x,y)$	0.8

(b) Relation probabilities

Figure A.6. Probabilistic event sequences for NoNeg2

A.1.3. NoNeg3

True	Unknown
$R(a,b)$	$P(c,b,a)$
$R(e,d)$	$Q(i,h)$
$S(d,e)$	$R(i,h)$
$T(a)$	$S(a,b)$
	$T(b)$

(a) A set (9 questions)

True		Unknown
$R(a,b)$	$T(a)$	$P(c,b,a)$
$R(e,d)$	$T(e)$	$Q(i,h)$
$R(g,f)$	$T(g)$	$R(i,h)$
$S(d,e)$	$T(h)$	$S(a,b)$
$S(f,g)$		$T(b)$

(b) B set (14 questions)

Figure A.7. Question sets for NoNeg3

1. P(x,y,z)	5. T(x)	9. T(x)
2. R(x,y)	6. ...	10. ...
3. T(x)	7. Q(y,x)	11. S(y,x)
4. ...	8. R(x,y), S(y,x)	12. ...

Figure A.8. Fixed event sequence for NoNeg3

Predicate	Prior
$P(x,y,z)$	0.2
$Q(y,x)$	0.2
$R(x,y)$	0.2
$S(y,x)$	0.5
$T(x)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x,y,z) \rightarrow R(x,y)$	0.75
$Q(y,x) \rightarrow R(x,y)$	0.9
$R(x,y) \rightarrow T(x)$	0.9
$Q(y,x) \rightarrow S(y,x)$	0.8

(b) Relation probabilities

Figure A.9. Probabilistic event sequences for NoNeg3

A.1.4. NoNeg4

True	Unknown
$Q(a,b)$	$P(c,d)$
$R(a)$	$Q(e,f)$
$S(a,b)$	$R(b)$
$S(g,h)$	$S(f,e)$
$T(b)$	$T(a)$
$V(g,h,i)$	$U(j,k,l)$
	$V(i,h,g)$

(a) A set (13 questions)

True		Unknown
$Q(a,b)$	$T(b)$	$P(c,d)$
$R(a)$	$T(d)$	$Q(e,f)$
$R(c)$	$T(h)$	$R(b)$
$S(a,b)$	$T(k)$	$S(f,e)$
$S(c,d)$	$T(f)$	$T(a)$
$S(g,h)$	$V(g,h,i)$	$U(j,k,l)$
$S(j,k)$		$V(i,h,g)$

(b) B set (20 questions)

Figure A.10. Question sets for NoNeg4

1. P(x,y)	6. Q(x,y)	11. T(y)	16. ...	21. ...	26. R(y)...
2. Q(x,y)	7. S(x,y), R(x)	12. ...	17. U(x,y,z)	22. V(x,y,z)	27. ...
3. S(x,y), R(x)	8. T(y)	13. T(y)	18. V(x,y,z)	23. S(x,y)	
4. T(y)	9. ...	14. ...	19. S(x,y)	24. T(y)	
5. ...	10. S(x,y)	15. R(x)	20. T(y)	25. ...	

Figure A.11. Fixed event sequence for NoNeg4

Predicate	Prior
$P(x, y)$	0.2
$Q(x, y)$	0.2
$R(x, y)$	0.5
$S(x, y)$	0.2
$T(y)$	0.2
$U(x, y, z)$	0.2
$V(x, y, z)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x, y) \rightarrow Q(x, y)$	0.9
$Q(x, y) \rightarrow R(x)$	0.8
$Q(x, y) \rightarrow S(x, y)$	0.75
$S(x, y) \rightarrow T(y)$	0.85
$U(x, y, z) \rightarrow V(x, y, z)$	0.7
$V(x, y, z) \rightarrow S(x, y)$	0.85

(b) Relation probabilities

Figure A.12. Probabilistic event sequences for NoNeg4

A.1.5. NoNeg5

True	Unknown
$R(a, b)$	$P(c, d)$
$S(b)$	$Q(c, d)$
	$R(e, f)$
	$S(a)$

(a) A set (6 questions)

True	Unknown
$R(a, b)$	$P(c, d)$
$S(b)$	$Q(c, d)$
$S(d)$	$R(e, f)$
	$S(a)$

(b) B set (7 questions)

True	Unknown
$P(a, b)$	$P(c, d)$
$Q(a, b)$	$Q(c, d)$
$R(c, d)$	$R(e, f)$
	$S(e)$

(c) C set (7 questions)

Figure A.13. Question sets for NoNeg5

1. P(x,y), Q(x,y)	5. R(x,y)	9. ...	13. ...
2. R(x,y)	6. S(y)	10. Q(x,y)	14. Q(x,y)
3. S(y)	7. ...	11. ...	15 ...
4. ...	8. P(x,y)	12. P(x,y)	

Figure A.14. Fixed event sequence for NoNeg5

Predicate	Prior
$P(x, y)$	0.2
$Q(x, y)$	0.2
$R(x, y)$	0.2
$S(y)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x, y) \wedge Q(x, y) \rightarrow R(x, y)$	0.9
$R(x, y) \rightarrow S(y)$	0.8

(b) Relation probabilities

Figure A.15. Probabilistic event sequences for NoNeg5

A.1.6. NoNeg6

True	Unknown
$Q(a, b)$	$P(d, e, g)$
$T(d, e)$	$Q(f, g)$
$T(f, g)$	$S(a, b)$
	$T(c, h)$

(a) A set (7 questions)

True	Unknown
$Q(a, b)$	$P(d, e, g)$
$T(a, b)$	$Q(f, g)$
$T(d, e)$	$S(a, b)$
$T(f, g)$	$T(c, h)$

(b) B set (8 questions)

True	Unknown
$P(a, b, c)$	$P(f, g, a)$
$Q(d, e)$	$Q(h, c)$
$R(a, b)$	$R(f, g)$
$R(d, e)$	$S(g, h)$
$S(f, g)$	$T(c, h)$

(c) C set (10 questions)

Figure A.16. Question sets for NoNeg6

1. P(x,y,z)	6. T(x,y)	11. P(x,y,z)	16. R(x,y)	21. ...	26 ...
2. Q(x,y), R(x,y)	7 ...	12. Q(x,y)	17. ...	22. R(x,y)	
3. T(x,y)	8. P(x,y,z)	13. ...	18. Q(x,y)	23. ...	
4. ...	9. Q(x,y)	14. Q(x,y)	19. ...	24. S(x,y)	
5. Q(x,y), R(x,y)	10. ...	15. ...	20. R(x,y)	25. T(x,y)	

Figure A.17. Fixed event sequence for NoNeg6

Predicate	**Prior**
$P(x, y, z)$	0.2
$Q(x, y)$	0.2
$R(x, y)$	0.2
$S(x, y)$	0.2
$T(x, y)$	0.2

(a) Prior probabilities

Relation	**Probability**
$P(x, y, z) \rightarrow Q(x, y)$	0.9
$Q(x, y) \wedge R(x, y) \rightarrow T(x, y)$	0.9
$S(x, y) \rightarrow T(x, y)$	0.8

(b) Relation probabilities

Figure A.18. Probabilistic event sequences for NoNeg6

A.1.7. NoNeg7

True	**Unknown**
$R(a, b)$	$P(e, f)$
$T(a, b)$	$Q(c, d)$
$U(c)$	$R(e, f)$
	$S(a)$
	$T(c, d)$
	$U(a)$

(a) A set (9 questions)

True	**Unknown**
$R(a, b)$	$P(e, f)$
$R(c, d)$	$Q(c, d)$
$T(a, b)$	$R(e, f)$
$U(c)$	$S(a)$
	$T(c, d)$
	$U(a)$

(b) B set (10 questions)

True	**Unknown**
$P(a, b)$	$P(e, f)$
$P(c, d)$	$Q(c, d)$
$Q(a, b)$	$R(e, f)$
$S(c)$	$S(a)$
	$T(c, d)$
	$U(a)$

(c) C set (10 questions)

Figure A.19. Question sets for NoNeg7

1. Q(x,y), R(x,y)	5. U(x)	9. ...	13. ...	17. ...	21. ...
2. T(x,y)	6. ...	10. R(x,y)	14. S(x)	18. S(x)	22. U(x)
3. ...	7. P(x,y)	11. ...	15. ...	19. ...	23 ...
4. R(x,y), S(x)	8. R(x,y)	12. Q(x,y)	16. Q(x,y)	20. T(x,y)	

Figure A.20. Fixed event sequence for NoNeg7

Predicate	**Prior**
$P(x, y)$	0.2
$Q(x, y)$	0.2
$R(x, y)$	0.2
$S(x)$	0.2
$T(x, y)$	0.2
$U(x)$	0.2

(a) Prior probabilities

Relation	**Probability**
$P(x, y) \rightarrow R(x, y)$	0.85
$Q(x, y) \wedge R(x, y) \rightarrow T(x, y)$	0.9
$R(x, y) \wedge S(x) \rightarrow U(x)$	0.8

(b) Relation probabilities

Figure A.21. Probabilistic event sequences for NoNeg7

A.1.8. NoNeg8

True	Unknown
$R(b,c)$	$P(e,g,h)$
$S(b)$	$Q(a)$
$U(e)$	$R(g,h)$
$V(h)$	$S(g)$
	$T(d,e)$
	$U(h)$
	$V(e)$

(a) A set (11 questions)

True	Unknown
$R(b,c)$	$P(e,g,h)$
$R(e,f)$	$Q(a)$
$S(b)$	$R(g,h)$
$S(h)$	$S(g)$
$U(b)$	$T(d,e)$
$U(e)$	$U(h)$
$V(b)$	$V(e)$
$V(h)$	

(b) B set (15 questions)

True	Unknown
$P(a,b,c)$	$P(e,g,h)$
$P(d,e,f)$	$Q(a)$
$Q(b)$	$R(g,h)$
$Q(h)$	$S(a)$
$S(e)$	$T(d,e)$
$T(a,b)$	$U(h)$
$T(g,h)$	$V(e)$

(c) C set (14 questions)

Figure A.22. Question sets for NoNeg8

1. P(x,y,z), Q(y)	7. ...	13. ...	19. ...	25. S(y)	31. T(x,y)
2. R(y,z), S(y)	8. S(y), T(x,y)	14. Q(y)	20. Q(y)	26. ...	32. ...
3. U(y)	9. V(y)	15. S(y)	21. S(y)	27. T(x,y)	33. U(y)
4. ...	10. ...	16. ...	22. ...	28. ...	34. ...
5. R(y,z), S(y)	11. P(x,y,z)	17. P(x,y,z)	23. R(y,z)	29. T(x,y)	35. V(y)
6. U(y)	12. R(y,z)	18. R(y,z)	24. ...	30. ...	36 ...

Figure A.23. Fixed event sequence for NoNeg8

Predicate	Prior
$P(x,y,z)$	0.2
$Q(y)$	0.2
$R(y,z)$	0.2
$S(y)$	0.2
$T(x,y)$	0.2
$U(y)$	0.2
$V(y)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x,y,z) \rightarrow R(y,z)$	0.85
$Q(y) \rightarrow S(y)$	0.9
$R(y,z) \wedge S(y) \rightarrow U(y)$	0.8
$S(y) \wedge T(x,y) \rightarrow V(y)$	0.85

(b) Relation probabilities

Figure A.24. Probabilistic event sequences for NoNeg8

A.2. Neg data sets

A.2.1. Neg1

True	False	Unknown
$Q(a,b)$	$R(c)$	$P(e,f)$ $R(a)$ $Q(c,d)$

Figure A.25. Question sets for Neg1 - A and B set (5 questions)

1. P(x,y)	4. ¬ P(x,y)
2. Q(x,y)	5. ¬ R(x)
3. ...	6 ...

Figure A.26. Fixed event sequence for Neg1

Predicate	Prior
$P(x, y)$	0.2
$\neg P(x, y)$	0.2
$Q(x, y)$	0.2
$\neg R(x)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x, y) \rightarrow Q(x, y)$	0.9
$\neg P(x, y) \rightarrow \neg R(x)$	0.9

(b) Relation probabilities

Figure A.27. Probabilistic event sequences for Neg1

A.2.2. Neg2

True	False	Unknown	
$S(a, b)$	$Q(a, b, c)$	$P(d, e, f)$	$R(a, b)$
	$R(e, f)$	$Q(a, b, f)$	$S(e, f)$

(a) A set (7 questions)

True	False	Unknown	
$S(a, b)$	$Q(a, b, c)$	$P(d, e, f)$	$R(a, b)$
$S(g, h)$	$R(e, f)$	$Q(a, b, f)$	$S(e, f)$

(b) B set (8 questions)

Figure A.28. Question sets for Neg2

1. P(x,y,z)	5. ¬ Q(x,y,z)	9. ...	13. ¬ R(y,z)	17. ...
2. ¬ Q(x,y,z)	6. S(x,y)	10. Q(x,y,z)	14. ...	18. ¬ R(y,z)
3. S(x,y)	7. ...	11. ¬ R(y,z)	15. Q(x,y,z)	19. ...
4. ...	8. S(x,y)	12. ...	16. ¬ R(y,z)	

Figure A.29. Fixed event sequence for Neg2

Predicate	Prior
$P(x, y, z)$	0.2
$Q(x, y, z)$	0.2
$\neg Q(x, y, z)$	0.2
$\neg R(y, z)$	0.2
$S(x, y)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x, y, z) \rightarrow \neg Q(x, y, z)$	0.9
$Q(x, y, z) \rightarrow \neg R(y, z)$	0.85
$\neg Q(x, y, z) \rightarrow S(x, y)$	0.85

(b) Relation probabilities

Figure A.30. Probabilistic event sequences for Neg2

A.2.3. Neg3

True	False	Unknown	
$S(a, b)$	$Q(b, a)$	$P(g, h)$	$S(g, h)$
$T(d)$	$R(c, d)$	$Q(h, g)$	$T(e)$
		$R(f, e)$	

(a) A set (9 questions)

True		False	Unknown	
$S(a, b)$	$T(d)$	$Q(b, a)$	$P(g, h)$	$S(g, h)$
$S(f, e)$	$T(h)$	$R(c, d)$	$Q(h, g)$	$T(e)$
			$R(f, e)$	

(b) B set (11 questions)

Figure A.31. Question sets for Neg3

1. P(x,y)	5. ¬ Q(y,x)	9. ...	13. ...	17. T(y)
2. ¬ Q(y,x)	6. S(x,y)	10. ¬ P(x,y)	14. ¬ R(x,y)	18. ...
3. S(x,y)	7. ...	11. ¬ R(x,y)	15. T(y)	
4. ...	8. S(x,y)	12. T(y)	16. ...	

Figure A.32. Fixed event sequence for Neg3

Predicate	Prior
$P(x, y)$	0.2
$\neg P(x, y)$	0.2
$\neg Q(y, x)$	0.5
$\neg R(x, y)$	0.5
$S(x, y)$	0.2
$T(y)$	0.2

(a) Prior probabilities

Relation	Probability
$P(x, y) \rightarrow \neg Q(y, x)$	0.9
$\neg P(x, y) \rightarrow \neg R(x, y)$	0.9
$\neg Q(y, x) \rightarrow S(x, y)$	0.85
$\neg R(x, y) \rightarrow T(y)$	0.8

(b) Relation probabilities

Figure A.33. Probabilistic event sequences for Neg3

A.2.4. Neg4

True	False	Unknown	
$Q(b, c)$	$T(f, g)$	$P(a, d, e)$	$S(f, g)$
$S(b, c)$	$U(b)$	$Q(k, l)$	$T(k, l)$
$S(j, h)$		$R(l, j, k)$	$U(f)$

(a) A set (11 questions)

True	False		Unknown	
$Q(b, c)$	$T(f, g)$	$U(j)$	$P(a, d, e)$	$S(f, g)$
$S(b, c)$	$U(b)$	$U(k)$	$Q(k, l)$	$T(k, l)$
$S(d, e)$	$U(d)$		$R(l, j, k)$	$U(f)$
$S(j, h)$				

(b) A set (15 questions)

Figure A.34. Question sets for Neg4

1.¬ P(x,y,z)	6. Q(y,z)	11. ¬ U(y)	16. ¬T(x,y)	21. ¬T(x,y)	26. S(y,z)
2. Q(y,z)	7. S(y,z)	12. ...	17. ...	22. ...	27. ¬ U(y)
3. S(y,z)	8. ¬ U(y)	13. ¬ U(y)	18. ¬T(x,y)	23. ¬T(x,y)	28 ...
4. ¬U(y)	9. ...	14. ...	19. ...	24. ...	
5. ...	10. S(y,z)	15. ¬Q(x,y)	20. ¬Q(x,y)	25. ¬ R(z,x,y)	

Figure A.35. Fixed event sequence for Neg4

Predicate	Prior
$\neg P(x, y, z)$	0.2
$Q(y, z)$	0.2
$\neg Q(y, z)$	0.2
$\neg R(z, x, y)$	0.2
$S(y, z)$	0.5
$\neg T(y, z)$	0.5
$\neg U(y)$	0.2

(a) Prior probabilities

Relation	Probability
$\neg P(x, y, z) \rightarrow Q(y, z)$	0.9
$Q(y, z) \rightarrow S(y, z)$	0.85
$\neg Q(y, z) \rightarrow \neg T(y, z)$	0.85
$\neg R(z, x, y) \rightarrow S(y, z)$	0.8
$S(y, z) \rightarrow \neg U(y)$	0.9

(b) Relation probabilities

Figure A.36. Probabilistic event sequences for Neg4

B. Learning and development stages

This appendix provides pseudocode for both stages of the learning and development cycle (algorithm 1, page 78).

B.1. Learning stage

Algorithm 2 Causal Hebbian Learning algorithm as implemented for the work presented in this thesis, based on description of causal Hebbian learning provided in [118]. The following occurs each time a new event observation is made.

```
Activate all nodes of predicate cluster corresponding to observed predicate
for each neuron i do
    for each neuron j that provides input to i do
        if i and j are collectors and j is active then
            if i is active but j was active first then
                Strengthen weight of connection from j to i
            end if
            if i was not active for the duration of the window of synchrony then
                Weaken weight of connection from j to i
            end if
        end if
        if i and j are enablers or role nodes and i is active then
            if j is active but i was active first then
                Strengthen weight of connection from j to i
            end if
            if j was not active for the duration of the window of synchrony then
                Weaken weight of connection from j to i
            end if
        end if
    end for
end for
```

B.2. Developmental stage

Algorithm 3 Development stage. The following occurs after an event observation has been made and the learning stage has been executed.

```
for each neuron i do
    Let N = root node of genome
    while type(N) = SELF do
        if condition specified in N is true for neuron i then
            Set N to genome node referenced by N.true
        else
            Set N to genome node referenced by N.false
        end if
    end while
    if type(N) = E_INPUT (existing input) then
        for each neuron j which provides input to i do
            while type(N) = E_INPUT do
                if condition specified in N is true for neuron j then
                    Set N to genome node referenced by N.true
                else
                    Set N to genome node referenced by N.false
                end if
            end while
            if type(N) = ACTION then
                Perform action specified by N on i, j pair
            end if
        end for
    end if
    if type(N) = P_INPUT (possible input) then
        for each neuron j which does not provide input to i do
            while type(N) = P_INPUT do
                if condition specified in N is true for neuron j then
                    Set N to genome node referenced by N.true
                else
                    Set N to genome node referenced by N.false
                end if
            end while
            if type(N) = ACTION then
                Perform action specified by N on i, j pair
            end if
        end for
    end if
end for
```

C. Variation operators

Algorithm 4 Mutation operation

```
Let pM = mutation rate
for each node N do
    if Node is a condition then
        mutateAttributeGene = newRandomNumber() < pM
        mutateOperatorGene = newRandomNumber() < pM
        mutateArgumentGene = newRandomNumber() < pM
        mutateTrueGene = newRandomNumber() < pM
        mutateFalseGene = newRandomNumber() < pM
        if mutateAttributeGene = True then
            Choose new value at random for 'attribute' gene
            if 'operator' gene is no longer compatible with new 'attribute' gene then
                mutateOperatorGene = True
            end if
            if 'argument' gene is no longer compatible with new 'attribute' gene then
                mutateAttributeGene = True
            end if
        end if
        if mutateOperatorGene = True then
            Choose new value at random for 'operator' gene
            if 'argument' gene is no longer compatible with new 'operator' gene then
                mutateAttributeGene = True
            end if
        end if
        if mutateArgumentGene = True then
            Choose new value at random for 'argument' gene
        end if
        if mutateTrueGene = True then
            Randomly choose new node to branch to if condition evaluates as 'true'
        end if
        if mutateFalseGene = True then
            Randomly choose new node to branch to if condition evaluates as 'false'
        end if
    end if
    if Node is an action then
        mutateActionGene = newRandomNumber() < pM
        mutateWeightGene = newRandomNumber() < pM
        if mutateActionGene = True then
            Choose new value at random for 'action' gene
        end if
        if mutateWeightGene = True then
            Choose new value at random for 'weight' gene
        end if
    end if
end for
```

Algorithm 5 Crossover operation

for i (index of parent) in [1, 2] **do**

Copy parent P_i to produce child C_i
Create U_i as a copy of all introns from P_i
Initialise $I_i = \{\}$ to later store subset of U_i
Create tr_i as a copy of a random exon tree from P_i

j = index of other parent
Create U_j as a copy of all introns from P_j
Initialise $I_j = \{\}$ to later store subset of U_j
Choose random exon tree tr_j from P_j such that $type(root(tr_i)) \preceq type(root(tr_j))$ and $size(tr_j) \leq size(tr_i \cup U_i)$ *

if $size(tr_i) < size(tr_j)$ **then**
  while $size(tr_i \cup I_i) < size(tr_j)$ **do**
    Remove random node from U_i and add to I_i
  end while
end if
if $size(tr_i) > size(tr_j)$ **then**
  while $size(tr_j \cup I_j) < size(tr_i)$ **do**
    if $size(U_j) > 0$ **then** Remove random node from U_j and add to I_j
    else Generate random node and add to I_j. †
    end if
  end while
end if

Remove tr_i and I_i from C_i and replace with tr_j and I_j in such a way that $root(tr_j)$ is placed in the original index of $root(tr_i)$.

end for

* If tr_j is larger than tr_i, it will replace a set of introns I_i in addition to tr_i. The condition $size(tr_j) \leq size(tr_i \cup U_i)$ guarantees that there are enough introns in C_i to be replaced.

† If tr_j is smaller than tr_i, tr_i will be replaced by a set of introns I_j in addition to tr_j. If there are insufficient introns in I_j for this purpose, additional introns are generated at random.

Bibliography

[1] J. B. Aimone, W. Deng, and F. H. Gage. Adult neurogenesis: integrating theories and separating functions. *Trends in cognitive sciences*, 14(7):325–337, 2010.

[2] J. R. Anderson. *Cognitive Psychology and its Implications*. W. H. Freeman and Company, 4th edition, 1995.

[3] J. Austin, S. Hobson, N. Burles, and S. O'Keefe. A rule chaining architecture using a correlation matrix memory. In *Artificial Neural Networks and Machine Learning - ICANN 2012*, pages 49–56. Springer Berlin Heidelberg, 2012.

[4] S. Bader and P. Hitzler. Dimensions of neural-symbolic integration - a structured survey. In S. N. Artëmov, H. Barringer, A. S. d'Avila Garcez, L. C. Lamb, and J. Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay*, pages 167–194. College Publications, 2005.

[5] S. Bader, P. Hitzler, and S. Hölldobler. Connectionist model generation: A first-order approach. *Neurocomputing*, 71(13):2420–2432, 2008.

[6] S. Bader, P. Hitzler, S. Hölldobler, and A. Witzel. A fully connectionist model generator for covered first-order logic programs. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-07*, pages 666–671, Hyderabad, India, 2007. AAAI Press.

[7] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic programming: an introduction*, volume 1. Morgan Kaufmann, 1998.

[8] G. Boella, S. C. Tosatto, L. V. D. Torre, A. S. d'Avila Garcez, and V. Genovese. Embedding normative reasoning into neural symbolic systems. In *Seventh International Workshop on Neural-Symbolic Reasoning and Learning*, pages 19–24, 2011.

[9] J. C. Bongard and R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001.

[10] A. Borji and L. Itti. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207, 2013.

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

[12] M. Brown, R. Keynes, and A. Lumsden. *The Developing Brain*. Oxford University Press, 2001.

[13] A. Chavoya. Artificial development. In *Foundations of Computational, Intelligence Volume 1*, pages 185–215. Springer, 2009.

[14] C. A. C. Coello, D. A. V. Veldhuisen, and G. B. Lamont. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[15] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning*. ACM, 2008.

[16] A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. Perspectives in Neural Computing. Springer, 2002.

[17] A. S. d'Avila Garcez and D. M. Gabbay. Fibring neural networks. In *Proceedings of 19th National Conference on Artificial Intelligence, AAAI'04*, pages 342–347, San Jose, CA, 2004.

[18] A. S. d'Avila Garcez and L. C. Lamb. Reasoning about time and knowledge in neural-symbolic learning systems. In S. B. Thrun, L. Saul, and B. Schoelkopf, editors, *Advances in Neural Information Processing Systems 16, Proceedings of NIPS 2003*, pages 921–928. MIT Press, 2003.

[19] A. S. d'Avila Garcez, L. C. Lamb, and D. M. Gabbay. Neural-symbolic intuitionistic reasoning. In A. Abraham, M. Köppen, and K. Franke, editors, *Design and application of hybrid intelligent systems*, volume 104 of *Frontiers in Artificial Intelligence and Applications*, pages 399–408, Amsterdam, 2003. IOS Press.

[20] A. S. d'Avila Garcez, L. C. Lamb, and D. M. Gabbay. Connectionist modal logic: Representing modalities in neural networks. *Theoretical Computer Science*, 371(1-2):34–53, 2007.

[21] A. S. d'Avila Garcez, L. C. Lamb, and D. M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Springer, 2008.

[22] R. Dawkins. *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design*. Norton, New York, 1986.

[23] H. de Garis. Artificial embryology and cellular differentiation. In *Evolutionary Design by Computers*, pages 281–295, San Francisco, USA, 1999. Morgan Kaufman Publishers.

[24] H. de Garis, M. Korkin, and G. Fehr. The CAM-brain machine (CBM). *Journal of Autonomous Robots*, 10, 2001.

[25] H. de Garis, J. Y. Tang, Z. Huang, L. Bai, C. Chen, S. Chen, J. Guo, X. Tan,

H. Tian, X. Tian, X. Wu, Y. Xiong, X. Yu, and D. Huang. The china-brain project: Building china's artificial brain using an evolved neural net module approach. In *Proceedings of the 2008 conference on Artificial General Intelligence*, pages 107–121, Amsterdam, The Netherlands, 2008. IOS Press.

[26] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[27] H. L. H. de Penning, R. J. M. den Hollander, H. Bouma, G. J. Burghouts, and A. S. d'Avila Garcez. A neural-symbolic cognitive agent with a mind's eye. In *Proceedings of AAAI Workshop on Neural-Symbolic Learning and Reasoning NeSy12*, pages 9–14, Toronto, Canada, 2012.

[28] H. L. H. de Penning, B. Kappé, and K. van den Bosch. A neural-symbolic system for automated assessment in training simulators. In *Proceedings of the Fifth International Workshop on Neural-Symbolic Learning and Reasoning (NeSy 09)*, pages 35–38, 2009.

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

[30] R. Eckhorn, H. J. Reitboeck, M. Arndt, and P. Dicke. Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex. *Neural Computation*, 2(3):293–307, 1990.

[31] P. 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.

[32] P. Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Proceedings of the 4th European Conference on Artificial Life (ECAL97)*, pages 205–213. MIT Press, 1997.

[33] S. W. Flavell and M. E. Greenberg. Signaling mechanisms linking neuronal activity to gene expression and plasticity of the nervous system. *Annual Review of Neuroscience*, 31:563–590, 2008.

[34] M. V. França, G. Zaverucha, and A. S. d'Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*, 94(1):81–104, 2014.

[35] V. Goel. Cognitive neuroscience of thinking. *Handbook of Neuroscience for the Behavioral Sciences*, 2009.

[36] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, volume 412. Addison-wesley, Reading Menlo Park, 1989.

[37] G. Gómez and P. E. Hotz. Investigations on the robustness of an evolved learning

mechanism for a robot arm. In *Proceedings of the 8th International Conference on Intelligent Autonomous Systems (IAS-8)*, pages 818–827, 2004.

[38] G. Gómez and P. E. Hotz. Evolutionary synthesis of grasping through self-exploratory movements of a robotic hand. In *Proceedings of the IEEE Congress on Evolutionary Computation 2007*, pages 3418–3425, 2007.

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

[40] B. Hammer and P. Hitzler. *Perspectives of Neural-Symbolic Integration*. Springer, 2007.

[41] D. O. Hebb. *The Organization of Behavior: A neuropsychological theory*. Wiley, New York, 1949.

[42] J. A. Hertz, A. S. Krogh, and R. G. Palmer. *Introduction To The Theory Of Neural Computation*. Studies in the Science of Complexity. Westview Press, Santa Fe Institute Series, 1991.

[43] C. J. Hogger. *Introduction to logic programming*. Academic Press Professional, Inc., 1984.

[44] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press, 1975.

[45] S. Hölldobler and Y. Kalinke. Towards a new massively parallel computational model for logic programming. In *Proceedings of the Workshop on Combining Symbolic and Connectionist Processing, ECAI 1994*, pages 68–77, 1994.

[46] S. Hölldobler, Y. Kalinke, and H. P. Storr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence Journal, Special Issue on Neural Networks and Structured Knowledge*, 11(1):45–58, 1999.

[47] P. E. Hotz, G. Gómez, and R. 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.

[48] J. E. Hummel and K. J. Holyoak. Distributed representations of structure: a theory of analogical access and mapping. *Psychological Review*, 104(3):427–466, 1997.

[49] J. E. Hummel and K. J. Holyoak. A symbolic-connectionist theory of relational inference and generalization. *Psychological review*, 110(2):220–264, 2003.

[50] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[51] H. Jacobsson. Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation*, 17(6):1223–1263, 2005.

[52] L. Kallel, B. Naudts, and A. Rogers, editors. *Theoretical aspects of evolutionary computing*. Springer, 2001.

[53] E. R. Kandel, J. H. Schwartz, and T. M. Jessel, editors. *Principles of neural science*. McGraw-Hill, New York, 4th edition, 2000.

[54] K. Kersting and L. De Raedt. Towards combining inductive logic programming with bayesian networks. In *Inductive Logic Programming*, pages 118–131. Springer, 2001.

[55] G. M. Khan, D. M. Halliday, and J. F. Miller. Intelligent agents capable of developing memory of their environment. *Advances in Modelling Adaptive and Cognitive Systems*, pages 77–114, 2010.

[56] H. 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.

[57] H. Kitano. A simple model of neurogenesis and cell differentiation based on evolutionary large-scale chaos. *Artificial Life*, 2(1):79–99, 1995.

[58] T. Kohonen. Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359, 1972.

[59] E. Komendantskaya and Q. Zhang. SHERLOCK - an interface for neuro-symbolic networks. pages 39–40. Proceedings of the Seventh International Workshop on Neural-Symbolic Learning and Reasoning (NeSy '11), 2011.

[60] A. Kumar, S. Rotter, and A. Aertsen. Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. *Nature Reviews Neuroscience*, 11(9), 2010.

[61] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

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

[63] R. P. Lippmann. Review of neural networks for speech recognition. *Neural Computation*, 1(1):1–38, 1989.

[64] W. Maass and C. M. Bishop, editors. *Pulsed neural networks*. MIT press, 2001.

[65] G. F. Marcus. Plasticity and nativism: Towards a resolution of an apparent paradox. In *Emergent neural computational architectures based on neuroscience*, pages 368–382. Springer, 2001.

[66] T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, 7(3), 1994.

[67] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

[68] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[69] J. McShane. *Cognitive development: An information processing approach.* Basil Blackwell, 1991.

[70] R. Miikkulainen. *Subsymbolic natural language processing: An integrated model of scripts, lexicon, and memory.* MIT press, 1993.

[71] J. F. Miller and G. M. Khan. Where is the brain inside the brain? *Memetic Computing*, 3(3):217–228, 2011.

[72] J. F. Miller and P. Thomson. Cartesian genetic programming. In *Proceedings of the 3rd European Conference on Genetic Programming*, volume 1802, pages 121–132, 2000.

[73] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[74] A. Mukerjee. Using attentive focus to discover action ontologies from perception. In *Proceedings of the Fifth International Workshop on Neural-Symbolic Learning and Reasoning (NeSy 09)*, pages 9–15, 2009.

[75] A. Mukerjee and M. M. Dabbeeru. Symbol emergence in design. In *Proceedings of the Fifth International Workshop on Neural-Symbolic Learning and Reasoning (NeSy 09)*, pages 29–34, 2009.

[76] A. Murray, editor. *Applications of neural networks.* Springer, 1994.

[77] M. Page. Connectionist modelling in psychology: A localist manifesto. *Behavioural and Brain Sciences*, 23(4):443–512, 2000.

[78] J. Piaget. Part I: Cognitive development in children: Piaget development and learning. *Journal of research in science teaching*, 2(3):176–186, 1964.

[79] S. Pinker. Kids say the darnedest things. In *Words and Rules: The Ingredients of Language*, pages 189–210. Weidenfeld and Nicholson, 1999.

[80] D. C. Plaut and J. L. McClelland. Locating object knowledge in the brain: Comment on bowers's (2009) attempt to revive the grandmother cell hypothesis. *Psychological Review*, 117(1), 2010.

[81] J. Prado, A. Chadha, and J. R. Booth. The brain network for deductive reasoning: A quantitative meta-analysis of 28 neuroimaging studies. *Journal of Cognitive Neuroscience*, 23(11), 2011.

[82] O. Ray and B. Golnia. A neural network approach for first-order abductive inference. pages 2–8. In Proceedings of the Fifth International Workshop on Neural-Symbolic Learning and Reasoning (NeSy 09), 2009.

[83] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[84] M. S. Rioult-Pedotti, D. Friedman, and J. P. Donoghue. Learning-induced LTP in neocortex. *Science*, 290(5491):533–536, 2000.

[85] S. Risi and K. O. Stanley. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artificial Life*, 18(4):331–363, 2012.

[86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986.

[87] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition, 1995.

[88] J. Secretan, N. Beato, D. B. D. Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley. Picbreeder: evolving pictures collaboratively online. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2008.

[89] L. Shastri. Advances in SHRUTI: A neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence Journal, Special Issue on Neural Networks and Structured Knowledge*, 11:79–108, 1999.

[90] L. Shastri. Types and quantifiers in SHRUTI - a connectionist model of rapid reasoning and relational processing. In *Hybrid Neural Systems*, pages 28–45. Springer Berlin Heidelberg, 2000.

[91] L. Shastri. Episodic memory trace formation in the hippocampal system: a model of cortico-hippocampal interactions. Technical report TR-01-004. Technical report, International Computer Science Institute, Berkeley, CA, USA, 2001.

[92] L. Shastri. From transient patterns to persistent structure: a model of episodic memory formation via cortico-hippocampal interactions. *Behavioral and Brain Sciences*, 2001.

[93] L. Shastri. A computationally efficient abstraction of long-term potentiation. *Neurocomputing*, 44:33–41, 2002.

[94] L. Shastri. Episodic memory and cortico-hippocampal interactions. *Trends in Cognitive Sciences*, 6:162–168, 2002.

[95] L. Shastri and V. Ajjanagadde. From simple associations to systematic reasoning: a connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16(03):417–494, 1993.

[96] L. Shastri and C. Wendelken. Learning structured representations. *Neurocomputing*, 52:363–370, 2003.

[97] N. T. Siebel and G. Sommer. Evolutionary reinforcement learning of artificial neural networks. *International Journal of Hybrid Intelligent Systems*, 4(3):171–183, 2007.

[98] A. Siegel and H. N. Sapru. *Essential Neuroscience*. Lippincott Williams & Wilkins, 2011.

[99] K. Sims. Evolving 3D morphology and behavior by competition. *Artificial Life*, 1(4):353–372, 1994.

[100] A. Slater and G. Bremner, editors. *An Introduction to Developmental Psychology: 2nd Edition*. John Wiley & Sons, 2011.

[101] S. G. Small and L. Medsker. Review of information extraction technologies and applications. *Neural Computing and Applications*, pages 1–16, 2013.

[102] J. P. Sougné. *INFERNET: A Neurocomputational Model of Binding and Inference*. PhD thesis, University of London, 1999.

[103] O. Sporns. Network analysis, complexity, and brain function. *Complexity*, 8(1):56–60, 2002.

[104] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machine*, 8(2):132–162, 2007.

[105] K. O. Stanley. Generative and developmental systems. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, 2009.

[106] K. O. Stanley, D. D'Ambrosio, and J. Gauci. Hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life Journal*, 15(2):185–212, 2009.

[107] K. O. Stanley and R. Miikkulainen. Efficient evolution of neural network topologies. In W. B. Langdon, E. Cantu-Paz, K. E. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, and A. C. Schultz, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1757–1762, Piscataway, NJ, 2002. San Francisco, CA: Morgan Kaufmann.

[108] K. Steinbuch. Die lernmatrix. *Biological Cybernetics*, 1(1):36–45, 1961.

[109] R. Sun. A tutorial on CLARION 5.0. 2003. http://www.cogsci.rpi.edu/~rsun/sun.tutorial.pdf.

[110] Y. Tang, E. Shimizu, G. R. Dube, C. Rampon, G. A. Kerchner, M. Zhuo, G. Liu, and J. Z. Tsien. Genetic enhancement of learning and memory in mice. *Nature*, 401(6748):63–69, 1999.

[111] G. G. Towell and J. W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1):119–165, 1994.

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

[113] J. Townsend, E. Keedwell, and A. Galton. Artificial development of connections in SHRUTI networks using a multi objective genetic algorithm. In *Proceedings of the fifteenth annual conference on Genetic and evolutionary computation conference companion*, Amsterdam, 2013. ACM.

[114] J. Townsend, E. Keedwell, and A. Galton. Evolution of connections in SHRUTI networks. In *Proceedings of the 9th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy13)*, Beijing, 2013.

[115] J. Townsend, E. Keedwell, and A. Galton. Artificial development of biologically plausible neural-symbolic networks. *Cognitive Computation*, 6(1):18–34, 2014.

[116] R. M. Twyman. *Instant Notes in Developmental Biology*. BIOS Scientific Publishers limited, Oxford, 2001.

[117] C. Wendelken. *SHRUTI-agent: A Structured Connectionist Architecture for Reasoning and Decision-Making*. PhD thesis, University of California, 2003.

[118] C. Wendelken and L. Shastri. Probabilistic inference and learning in a connectionist causal network. In *Proceedings of the Second International Symposium on Neural Computation*, pages 26–28, 2000.

[119] C. Wendelken and L. Shastri. Combining belief and utility in a structured connectionist agent architecture. In *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society*, 2002.

[120] C. Wendelken and L. 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.

[121] A. Wichert. Neural sub-symbolic reasoning. In *Seventh International Workshop on Neural-Symbolic Learning and Reasoning*, pages 2–7, 2011.

[122] B. G. Woolley and K. O. Stanley. On the deleterious effects of a priori objectives on evolution and representation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, New York, NY, 2011. ACM.

[123] C. Zhao, W. Deng, and F. H. Gage. Mechanisms and functional implications of adult neurogenesis. *Cell*, 132(4):645–660, 2008.