Abductive Reasoning in Neural-Symbolic Systems

Artur d'Avila Garcez, Dov Gabbay, Oliver Ray and John Woods

Abstract

Abduction is or subsumes a process of inference. It entertains possible hypotheses and it chooses hypotheses for further scrutiny. There is a large literature on various aspects of non-symbolic, subconscious abduction. There is also a very active research community working on the symbolic (logical) characterisation of abduction, which typically treats it as a form of hypothetico-deductive reasoning process. In this paper we start to bridge the gap between the symbolic and non-symbolic approaches to abduction. We are interested in benefiting from developments made by each community. In particular, we are interested in the ability of non-symbolic systems (neural networks) to learn from experience using efficient algorithms and to perform massively parallel computations of alternative abductive explanations. At the same time, we would like to benefit from the rigour and semantic clarity of symbolic logic. We present two approaches to dealing with abduction in neural networks. One of them uses Connectionist Modal Logic and a translation of Horn clauses into modal clauses to come up with a neural network ensemble that computes abductive explanations in a top-down fashion. The other combines neural-symbolic systems and abductive logic programming and proposes a neural architecture which performs a more systematic, bottom-up computation of alternative abductive explanations. Both approaches employ standard neural network architectures which are already known to be highly effective in practical learning applications. Differently from previous work in the area, our aim is to promote the integration of reasoning and learning in a way that the neural network provides the machinery for cognitive computation, inductive learning and hypothetical reasoning, while logic provides the rigour and explanation capability to the systems, facilitating the interaction with the outside world. Although it is left as future work to determine whether the structure of one of the proposed approaches is more amenable to learning than the other, we hope to have contributed to the development of the area by approaching it from the perspective of symbolic and sub-symbolic integration.

1 Introduction

Three notable hallmarks of intelligent cognition are the ability to draw rational conclusions, the ability to make plausible assumptions, and the ability to generalise from experience. In a logical setting, these abilities correspond to the processes of deduction, abduction, and induction, respectively. Although human cognition often involves the interaction of these three abilities, they are typically studied in isolation. For example, in Artificial Intelligence (AI), symbolic (logic-based) approaches have been mainly concerned with deductive reasoning, while connectionist (neural networks-based) approaches have focused on inductive learning. It is well known this connectionist/symbolic dicothomy in AI reflects a distinction between brain and mind, but, even so, we argue this should not disuade us from seeking a fruitful synthesis of these paradigms. As remarked by Smolensky, “if we believe that a mind is an abstract, higher-level description of a brain... and if we believe that connectionist networks provide a useful stand-in for a solid theory of neural computation... then it follows that abstract, higher-level descriptions of connectionist computation should provide the basis for theories of mind” [Smolensky, 2000].

In our research programme, we seek to integrate the processes of abduction, induction and deduction within the neural computation paradigm. Our goal is to develop a unified framework for learning and reasoning that exploits the parallelism and robustness of connectionist architectures. To this end, we choose to work with standard neural networks whose learning capabilities have already been demonstrated in significant practical applications [Rumelhart et al., 1986], and investigate how they can be enhanced with more advanced reasoning capabilities [d’Avila Garcez et al., 2002, d’Avila Garcez et al., 2006]. In this enterprise, we take inspiration from recent work in symbolic AI [Flach and Kakas, 2000, Ray, 2005], which shows the benefits that can arise from incorporating abductive reasoning within inductive learning. We also take inspiration from work in connectionist reasoning, which provides added reasoning capabilities to neural learning systems and a bridge between the symbolic and neural settings of computation [Smolensky and Legendre, 2006, Sun, 1995].

In neural computation, induction is typically seen as the process of changing the weights of a network in ways that reflect the statistical properties of a dataset (set of examples), allowing for useful generalisations over unseen examples. In the same setting, deduction is the network computation of output values as a response to input values (stimuli), given a particular set of weights. Such network computations have been shown equivalent to different logical systems depending on the network architecture (see e.g. [d'Avila Garcez et al., 2002]). Abduction, in this setting, can be seen as the process of finding input values that would result in a particular output value if presented to the network. In this process, we can either hypothesise input values and find out the network's outputs for those, or we can try and reverse the network's reasoning process so that the possible inputs for the desired output values can be computed. We shall investigate both approaches in this paper.

When we combine deduction and induction in neural networks, we use the same network to learn a current state of affairs and to compute its logical consequences. As we propose to combine abduction and induction in neural networks, we envisage a system where the possible outcomes of a current state of affairs can be considered (abduced), and inform a learning or belief revision process. In this paper, we focus on how neural networks can be used for abductive reasoning. The details of the interaction between abduction and learning are left as future work. It also remains to investigate how neural networks could be used in practice to combine all three primary inference processes.

Differently from previous work combining the symbolic and neural views of cognition, we choose to consider an artificial neural network model that has been shown very effective already in learning applications - and which we have already shown capable of deductive reasoning of various kinds, including nonmonotonic, modal and intuitionistic reasoning [d'Avila Garcez et al., 2002, d'Avila Garcez et al., 2006, d'Avila Garcez et al., 2006] - and we investigate how we can incorporate abduction into this system. The artificial neural network model in question consists of feedforward and recurrent networks, as opposed to the symmetric networks investigated e.g. in [Smolensky, 2000]. It uses a localist rather than a distributed representation [Page, 2000], and it works well with the highly effective Backpropagation neural learning algorithm [Rumelhart et al., 1986]. We will get back to issues of representation and learning later on in the paper.

We now concentrate on the problem of abduction. In its barest form, abduction is a reaction of a certain kind to a cognitive irritant. The irritation is occasioned by the inability to hit some cognitive target with present epistemic resources. The cognitive target is in its turn constituted by some or other state of affairs. The target may be a perplexing observation for which a plausible or rational explanation is sought, or it may represent a desirable goal which an agent would like a plan to achieve.

Abduction is, or even subsumes, a process of inference. It entertains possible hypotheses and it chooses hypotheses for further scrutiny. As such, abductive conclusions are not matters for belief or for probability, but they are mere suggestions that, if true, would offer an explanation for the cognitive target. Peirce famously characterised abduction as an inference of the form: the surprising fact C is observed; but if A were true, C would be a matter of course. Hence, there is reason to suspect (albeit tentatively) that A is true.

There is a large literature - if not a large consensus - on various aspects of non-symbolic, subconscious abduction. As Churchland observes, "...one understands at a glance why one end of the kitchen is filled with smoke: the toast is burning!" Churchland proposes that in matters of perceptional understanding, we possess "an organised library of internal representations of various perceptual situations, situations to which prototypical behaviours are the computed output of the well-trained network". Like Peirce, Churchland sees perception as a limit of explanation, and he suggests that all types of explanation can be modelled as prototype activation by way of "...vector coding and vector-to-vector transformation", rather than linguistic representation and standardly logical reasoning. In this approach the knowledge that comes from experience (learning) is modelled in the patterning of weights in the subject's neural network, where it is seen as a disposition of the system to assume various activation configurations in the face of various inputs [Churchland, 1989].

This form of abductive reasoning has been studied in the context of diagnostic problem solving - where outputs, called *manifestations*, are each associated with a set of possible inputs, called *disorders*. Given a set of manifestations, the abductive task is to find a set of disorders that account for, or *cover*, the manifestations according to the specified causal associations. Typically, the diagnoses must satisfy some notion of *parsimony*, which in its simplest form means that no subset of the disorders suffices to explain the manifestations.

A number of non-symbolic (neural network) models for diagnostic problem solving have been proposed. For example, [Goel and Ramanujam, 1996] proposes to re-write such diagnostic covering problems as constrained optimisations that are solved by energy minimisation in Hopfield networks, while [Reggia et al., 1993] and [Ayeb et al., 1998] apply similar methods to harder problems using competition-based neural networks. An extension of these methods is proposed in [Zhang and Xu, 1999] to enable abduction in more complex causal networks. While some of these approaches have been applied to real problems, they are limitied in both their expressivity and their ability to systematically compute alternative solutions.

These probelms are overcome in the symbolic setting, where abduction is most commonly treated as a general form of hypothetico-deductive reasoning which returns a set of facts Δ that explain a set of goals G with respect to a prior logical theory T. An account of the processes of abduction in the form of symbol manipulation can be found in [Levesque, 1989]. Logically, given a theory T and goals G, the task of abduction is to find a hypothesis Δ such that $T \cup \Delta$ logically entails G. Typically, one considers simple theories expressed in the form of Horn clauses or logic programs.

In this paper we start to bridge the gap between the symbolic and non-symbolic approaches to abduction. We are interested in benefiting from developments made by each research community. When we think of neural networks, what springs to mind is their ability to learn from examples using efficient algorithms in a massively parallel fashion. When we think of symbolic logic, we would like to benefit from the rigour, neat definitions, and semantic clarity it offers, or available proof procedures and the explanations they can give to the reasoning process, e.g. in the form of a proof history.

In particular, this paper seeks to bridge the gap between the processes of abductive reasoning and those of learning by experience. Differently from previous work in this respect [Flach and Kakas, 2000, Michalski, 1993], our aim is to develop massively parallel techniques for abduction that can be integrated with existing connectionist learning approaches. In this view, learning is achieved by generalisations over vector-to-vector transformations, presented as training examples to neural networks, and abductive inference is the process of computing neural activations that can be seen, in a principled way, as explanations for certain other neural activations. The neural network provides the machinery for cognitive computation, inductive learning and hypothetical reasoning, while logic provides the rigour and explanation capability to the models, facilitating the interaction with the outside world. We call such a system, combining a connectionist component with a logical component, a *neural-symbolic learning system* [d'Avila Garcez et al., 2002].

We consider two approaches in this paper. One of them is a top-down approach based on translating modal logic theories into ensembles of neural networks [d'Avila Garcez et al., 2006, d'Avila Garcez et al., 2004]. We investigate how this approach, called *Connectionist Modal Logic* (CML), can be used for abductive reasoning. The other approach is a bottom-up approach based on generalising neural-symbolic systems from logic programs to Abductive Logic Programs [Kakas et al., 1992] in order to systematically computate alternative abductive explanations and allow reasoning with partial information and integrity constraints.

In what follows, we briefly recall Neural-Symbolic Learning Systems (Section 2). We then review Connectionist Modal Logic and show how it can be used for abductive reasoning (Section 3). We then review Abductive

Logic Programming and show how it can be encoded in neural networks (Section 4). We conclude by summarising our case for the combination of symbolic and non-symbolic abduction, and by discussing directions for future work.

2 Neural-Symbolic Learning Systems

In this section, we define neural networks and present the basic concepts of neural-symbolic learning systems used throughout the paper.

An artificial neural network is a directed graph with the following structure: a node (or neuron) in the graph is characterised, at time t, by its *input vector* $I_i(t)$, its *input potential* $U_i(t)$, its *activation state* $A_i(t)$, and its *output* $O_i(t)$. The nodes of the network are interconnected via a set of directed and weighted edges (or connections) such that if there is a connection from node i to node j then $W_{ji} \in \mathbb{R}$ denotes the *weight* of this connection. The input potential $U_i(t)$ of neuron i at time t is obtained by computing a weighted sum for neuron i such that $U_i(t) = \sum_j W_{ij} I_i(t)$. The activation state $A_i(t)$ of neuron i at time t - a bounded real or integer number - is then given by the neuron's *activation function* h_i such that $A_i(t) = h_i(U_i(t))$. Typically, h_i is either a linear function, a non-linear (step) function, or a sigmoid function (e.g.: $tanh(x)$). In addition, θ_i (an extra weight with input always fixed at 1) is known as the *threshold* of neuron i. We say that neuron i is *active* at time t if $A_i(t) > \theta_i$. Finally, the neuron's output value $O_i(t)$ is given by its output function $f_i(A_i(t))$. Usually, f_i is the identity function.

The nodes of a neural network can be organised in layers. A *n-layer* *feedforward network* is an acyclic graph. It consists of a sequence of layers and connections between successive layers, containing one input layer, $n-2$ hidden layers, and one output layer, where $n \geq 2$. When $n = 3$, we say that the network is a *single hidden layer network.* When each node occurring in the *i-th* layer is connected to each node occurring in the $i+1$-*st* layer, we say that the network is *fully-connected.*

A multilayer feedforward network computes a function $\varphi : \mathbb{R}^r \rightarrow \mathbb{R}^s$, where r and s are the number of neurons occurring, respectively, in the input and output layers of the network. In the case of single hidden layer networks, the computation of φ occurs as follows: at time t_1, the input vector is presented to the input layer. At time t_2, the input vector is propagated through to the hidden layer, and the neurons in the hidden layer update their input potential and activation state. At time t_3, the hidden layer activation state is propagated to the output layer, and the neurons in the output layer update their input potential and activation state. At time t_4, the output vector is read off the output layer. In addition, most neural models have a *learning rule*, responsible for changing the weights of the network progressively so that it learns to approximate φ given a number of *training examples* (input vectors and their respective target output vectors).

In the case of *backpropagation* - the neural learning algorithm most successfully applied in industry [Rumelhart et al., 1986] - an *error* is calculated as the difference between the network's actual output vector and the target vector, for each input vector in the set of examples. This error $\mathbf{E}$ is then propagated back through the network, and used to calculate the variation of the weights $\triangle\mathbf{W}$. This calculation is such that the weights vary according to the *gradient* of the error, i.e. $\triangle \mathbf{W} = -\eta \nabla \mathbf{E}$, where $0 < \eta < 1$ is called the *learning rate*. The process is repeated a number of times in an attempt to minimise the error, and thus approximate the network's actual output to the target output, for each example. In order to try and avoid shallow local minima in the error surface, a common extension of the learning algorithm above takes into account, at any time t, not only the gradient of the error function, but also the variation of the weights at time $t-1$, so that $\triangle \mathbf{W}_t = -\eta \nabla \mathbf{E} + \mu \triangle \mathbf{W}_{t-1}$, where $0 < \mu < 1$ is called the *term of momentum*. Typically, a subset of the set of examples available for training is left out of the learning process so that it can be used for checking the network's *generalisation* ability, i.e. its ability to respond well to examples not seen during training.

Neural-Symbolic Learning Systems [d'Avila Garcez et al., 2002] are massively parallel computational models based on artificial neural networks that integrate inductive learning by backpropagation with deductive reasoning. In such systems, a *translation algorithm* maps a logical theory T into a single hidden layer neural network N such that N computes the deductive consequences of T. This should provide not only a massively parallel model for computing T, but one should be able to train N by examples, using T as background knowledge in a process of knowledge acquisition. The knowledge acquired by training could then be extracted closing the learning cycle, as advocated in [Towell and Shavlik, 1994].

In this paper, we consider theories T that are sets of Horn clauses each of the form $a_1, \ldots, a_n \rightarrow a_0$, where the a_i are ground atoms, a_0 is referred to as the *head* atom of the clause and $a_1, \ldots, a_n$ are called *body* atoms. Intuitively, this rule states that "if all of the body atoms $a_1, \ldots, a_n$ are true, then the head atom a_0 must also be true". If the body is empty, then the head is called a *fact* and is simply written a_0. A set of such clauses is usually called a (propositional) *Horn theory* or *logic program* [Lloyd, 1987].

Example 2.1 *Consider the neural-symbolic learning system of Figure 1. It encodes the logic program $P = \{r_1 : a, b \rightarrow x;\ r_2 : c \rightarrow x;\ r_3 : x \rightarrow y\}$*[1]*. Each rule r_i of P is mapped from the neural network's input layer to its output layer through a unique hidden neuron n_i. The output is activated if the rule's body is satisfied. For example, output neuron x will be activated (indicating that atom x is true) either if input neurons a and b are both activated (indicating that a and b are true), or if neuron c is activated (c is true); y will be activated if x is activated. In other words, the network computes an* and-or *function with hidden neurons computing a logical* and, *and output neurons computing a logical* or. *In addition, atoms that appear in the head of one rule and the body of another (e.g. x in r_1 and r_3) are linked through a feedback connection with weight fixed at 1.0 from the output to the input. This is responsible for implementing* chains *such as $a \rightarrow b$ and $b \rightarrow c$ in the network. In the case of P, this is how, given a and b, the network would have y activated via neuron x. The details of how to set-up the weights and thresholds of the network so that the appropriate behaviour of the logic is achieved can be found in [d'Avila Garcez et al., 2002].*

Given that neural-symbolic translations produce neural networks that represent a logical theory, and taking the view that abduction is a form of reverse deduction, it might be suspected that abductive reasoning could be achieved by simply running the network in reverse (i.e reasoning backwards from the desired output to the required inputs). However, since neural-symbolic systems exploit the massive parallelism of neural networks, this strategy would not work, as the following example illustrates.

Example 2.2 *Consider again the neural-symbolic learning system of Figure 1. Taking the task of symbolic abduction described above, we know that* $\{a, b\}$ *should be a possible explanation for* x *and so should* $\{c\}$*. If we were to simply reverse the network in an attempt to compute explanations* $\{a, b\}$ *and* $\{c\}$ *given hypothesis* x*, we would have a relation to compute instead of a function. As a result, a standard neural network (which computes functions, and not relations) would not be able to distinguish* $\{a, b\}$ *and* $\{c\}$ *as two alternative explanations for* x*. Instead, neurons* a, b *and* c *would be activated given* x *with no possible distinction between them.*

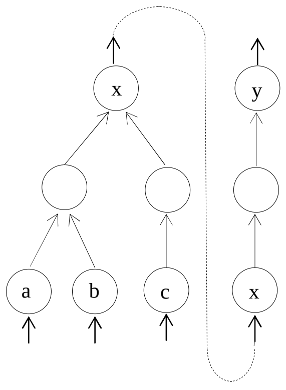

Fig. 1: A Neural-Symbolic Learning System

Notice that learning from examples can be performed on the network of Figure 1 to change the causal dependencies that have been pre-established by the background knowledge between the concepts in the input and output layers. For example, learning may cause the weights of the connections from neurons a and b to change so that, e.g., a alone (instead of a and b together) causes x. Notice further that, in general, the concepts to be considered $a, b, c, x, y, ...$ do not change. In other words, the network architeture is fixed. Future extensions of the learning system could, however, consider actual changes in architecture, which may be of interest in relation to the creation of wholly new hypotheses in the context of abduction.

In what follows, we assume that abduction presupposes an underlying deductive relation. We argue that, as a result, neural-symbolic learning systems should be a natural candidate for sub-symbolic abduction. We seek to be able to exploit the massive parallelism and the learning capabilities of neural systems, and our key challenge in this undertaking lies in the inevitable plurality of solutions yielded by abductive reasoning. In what follows, we propose two alternative solutions to the problem of sub-symbolic abduction, one based on the fact that connectionist modal logic can compute relations (this is a top-down approach), and the other based on fixed-point computations by a neural network of an abductive logic program meaning operator (a bottom-up approach).

3 Neural-Symbolic Abductive Reasoning

We investigate two solutions to the problem of realising abductive inference using Neural-Symbolic Learning Systems. As mentioned above, the key challenge faced in this undertaking is the inevitable plurality of solutions yielded by abduction. Abduction can only suggest tentative hypotheses that would, if they were true, explain a goal. We are therefore confronted with a situation where many alternative possible states of affairs must be considered. Our first solution addresses this issue by means of propositional modal logic with its associated possible world Kripke semantics [Hughes and Cresswell, 1996]. Our second solution is based on generalising existing neural-symbolic translations from normal logic programs to the case of abductive logic programs [Kakas et al., 1994], which are more expressive than normal programs and allow reasoning with incomplete information. The first approach has been outlined in [Gabbay and Woods, 2005], and the second in [Ray and dAvila Garcez, 2006].

Take the network shown in Figure 1 and assume that y is a goal. We would like to be able to reason top-down in order to obtain $\{x\}$ as an explanation for y, and then $\{a, b\}$ and $\{c\}$ in parallel, as alternative explanations for x. One way to achieve this is to use *Connectionist Modal Logic* (CML) [d'Avila Garcez et al., 2006] as detailed below.

Modal logic deals with the analysis of concepts such as necessity (represented by $\Box$) and possibility (represented by $\Diamond$). A key aspect of modal logic is the use of *possible worlds* and a binary (accessibility) relation R between possible worlds. In modal logic, a proposition is necessary in a world if it is true in all worlds which are possible in relation to that world, whereas it is possible in a world if it is true in at least one world which is possible in relation to that same world.

In Connectionist Modal Logic, we use ensembles of neural networks (instead of single networks) to represent the language of modal logic programming [Orgun and Ma, 1994]. The theories T are now sets of modal Horn clauses each of the form $Ma_1, \dots, Ma_n \rightarrow Ma_0$, where $M \in \{\Box, \Diamond\}$ and the a_i are ground atoms as before. Such theories can be implemented in neural-symbolic learning systems with the use of an ensemble of neural networks, each network representing a possible world. The use of an ensemble allows for the representation of relations (such as the accessibility relation of modal logic) in neural networks and thus provides a way of distinguishing $\{a, b\}$ and $\{c\}$ in the example above, as two alternative explanations for x.

In CML, each network in the ensemble is a simple single hidden layer network like the network of Figure 1 to which standard neural learning algorithms can be applied. Learning, in this setting, can be seen as learning the concepts that hold in each possible world independently, with the assessibility relation providing the information on how the networks should interact. For example, take three networks all related to each other. If neuron $\Diamond a$ is activated in one of these networks then a neuron a must be activated in at least one of the other networks. If neuron $\Box a$ is activated in one network then neuron a must be activated in all the networks. This implements in a connectionist setting the possible world semantics mentioned above; it can be achieved by defining the connections and the weights of a network ensemble, following a translation algorithm for modal Horn clauses which extends that for Horn clauses. Details of the translation using practical examples can be found in [d'Avila Garcez et al., 2004].

In the case of abductive reasoning, we can model, for example, the fact that $\{a, b\}$ and $\{c\}$ are possible explanations for x by having neurons a and b active in a network of the ensemble (or world) (say, w_1), and neuron c also active but in a different network of the ensemble (say, w_2) [2]. We denote these possibilities by $\Diamond(a,b)$ and $\Diamond(c)$, respectively. Neurons a and b should be active in w_1 whenever neuron x is active in a network w that is related to w_1. Similarly, neuron c should be active in w_2 whenever x is active in w. We denote the relationship between networks by the accessibility relation R so that in this case we have $R(w,w_1)$ and $R(w,w_2)$. In general, we will use $a_0 \rightarrow \Diamond(a_1,...,a_n)$ to denote the fact that $(a_1,...,a_n)$ is a possible explanation for a_0. The following example illustrates the idea.

Example 3.1 *Take the same program $P = \{r_1 : a,b \rightarrow x; r_2 : c \rightarrow x; r_3 : x \rightarrow y\}$. First, we translate P into a modal program by replacing each rule of the form $a_1,...,a_n \rightarrow a_0$ by a modal rule of the form $a_0 \rightarrow \Diamond(a_1,...,a_n)$. The intuition behind this translation is that $a_1,...,a_n$ is a possible explanation for a_0 (and hence the use of $\Diamond$). We then need to assign worlds to rules and define how the worlds should relate to each other (i.e. define the accessibility relation $R(w_i,w_j)$ between worlds w_i and w_j). We do so according to the* dependency chains *in P. For example, there is a dependency chain from rule r_3 to rule r_1 because the head of r_1 (i.e. x) forms part of the body of r_3. In this case, we must relate r_3 to r_1 so that when x is identified as an explanation for y, a,b can be identified as an explanation for x; if r_1 is labelled by world w_1, and r_3 is labelled by w_2, we make $R(w_2,w_1)$.*

Once we have a modal program and accessibility relation, an ensemble of neural networks can be constructed with the help of CML. For each possible world w_i, CML repeats the process for constructing single *and-or* networks, and then whenever $R(w_i,w_j)$, it connects each neuron $\Diamond(a_1,...,a_n)$ in w_i to neurons a_1 to a_n in w_j, as the following example illustrates.

Example 3.2 *Figure 2 shows the network ensemble* $w_1, ..., w_4$ *for the program* P *of Example 3.1. To each head atom (*x *and* y*), we assign a world (*w_1 *and* w_2*, respectively), and translate the program into the modal program:* $w_1 : x \rightarrow \Diamond(a, b); w_1 : x \rightarrow \Diamond c; w_2 : y \rightarrow \Diamond x$*. From the dependencies in* P*, we know that* $R(w_2, w_1)$*. This deals with* $\Diamond x$ *in* w_2 *(it tells us that* $\Diamond x$ *should be connected to* x *in* w_1*). Now, we need to deal with* $\Diamond(a, b)$ *and* $\Diamond c$ *in* w_2*. CML does this by creating possible worlds* w_3 *and* w_4 *s.t.* $R(w_2, w_3)$ *and* $R(w_2, w_4)$*. It then connects neuron* $\Diamond(a, b)$ *in* w_2 *to neurons* a *and* b *in* w_3 *so that* a *and* b *are activated whenever* $\Diamond(a, b)$ *is activated, and it connects neuron* $\Diamond c$ *in* w_2 *to neuron* c *in* w_4 *so that* c *is activated whenever* $\Diamond c$ *is activated. In the network of Figure 2, if at time* t *we activate* x*, at time* $t + 4$*,* a*,* b *and* c *will be activated. Similarly, if* y *is activated at* t *then* x *will be activated at* $t + 4$*. This allows us to reason top-down in parallel, at the same time keeping track of the alternative explanations for our hypotheses.*

In summary, given a set P of Horn clauses $a_1, ..., a_n \rightarrow a_i$, we proceed as follows: (i) create a world w_i for each head atom a_i in P; (ii) create a modal rule $w_i : a_i \rightarrow \Diamond(a_1, ..., a_n)$ for each clause in P; (iii) make $R(w_j, w_i)$ whenever a_i is in the body of a clause in P whose head is a_j; (iv) build a network ensemble for the modal rules using the CML translation algorithm [d'Avila Garcez et al., 2004]; (v) for each unconnected output neuron of the form $\Diamond(a_1, ..., a_n)$ in the ensemble, create a new network with output neurons $a_1, ..., a_n$, and connect $\Diamond(a_1, ..., a_n)$ to $a_1, ..., a_n$ so that $a_1, ..., a_n$ is activated if $\Diamond(a_1, ..., a_n)$ is activated; (vi) connect each output neuron a_i

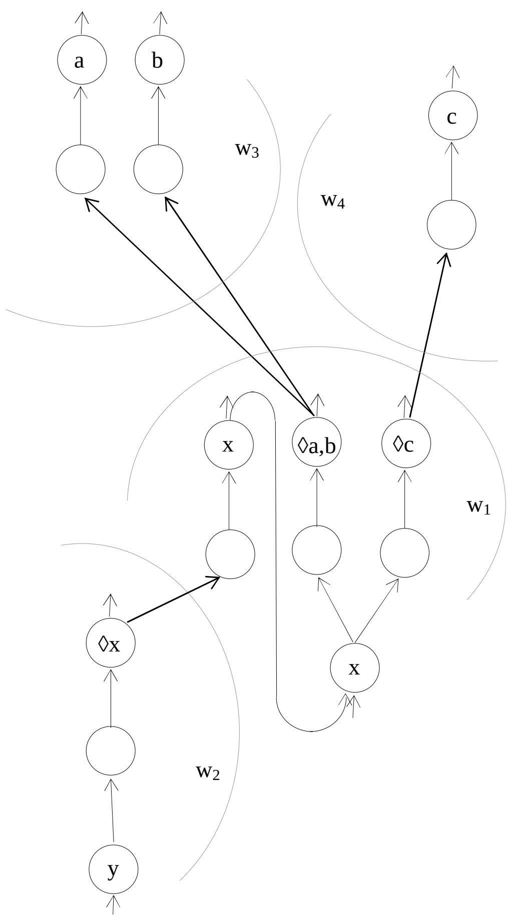

Fig. 2: A Neural Network Ensemble for Abductive Reasoning

in the ensemble to any corresponding output neuron with the same name (a_i) in any other network of the ensemble such that one neuron is activated whenever the other is activated. In order to run the network ensemble to compute explanations, we activate a number of neurons (hypotheses) at time t_1 and check the activation throughout the ensemble at times t_2, t_3, etc. until the ensemble is stable (i.e. until the activation at time t_i is the same as the activation at time t_{i-1}). The sequence of activations should give us the alternative explanations for the hypotheses, as exemplified above.

It is interesting to note the need for step (vi) in the above algorithm. This is related to the fact that, in some cases, different copies of the same atom will appear throughout the ensemble. Take, for example, the program: $a, b \to x; a \to y; b \to y; z \to b$. Our translation gives the modal program: $w_1 : x \to \Diamond(a, b); w_2 : y \to \Diamond a; w_2 : y \to \Diamond b; w_3 : b \to \Diamond z$, and the relation $R(w_2, w_3)$. We then create networks w_1, w_2 and w_3 as depicted in Figure 3. $R(w_2, w_3)$ tells us to connect $\Diamond b$ in w_2 to b in w_3. But $\Diamond(a, b)$ in w_1, $\Diamond a$ in w_2, and $\Diamond z$ in w_3 still need to be resolved. For this, we need to create three new networks w_4, w_5 and w_6 in which to place, respectively, a, a and b, and z. Figure 3 shows w_4 and w_5, but omits w_6 for simplicity. Notice how we are left with two copies of a and two copies of b in the ensemble. This is required because we need to be able to distinguish $\{a\}$ and $\{b\}$ as alternative explanations for y from $\{a, b\}$ as an explanation for x. Now, regardless of whether b is activated in w_3 or in w_5, we know that z should be activated in w_6. Similarly, $w_4 : a$ and $w_5 : a$ should activate each other. In order to achieve this, we connect such neurons via a hidden neuron in each network of the ensemble as shown in Figure 3 for neurons a and b. This creates a self-sustaining loop such that the activation of any such neuron causes the activation of the other and vice-versa, implementing the desired effect in the network ensemble. This is the purpose of step (vi) in the above algorithm.

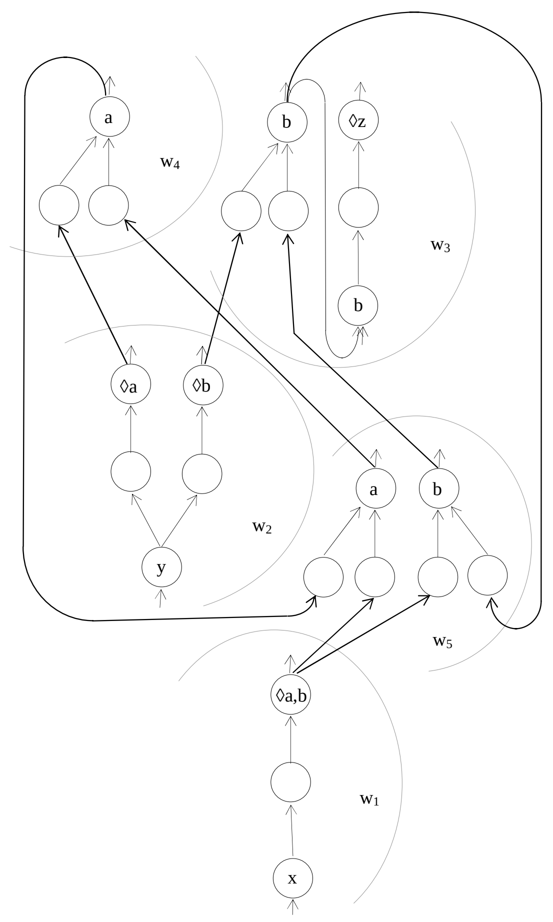

Fig. 3: Dealing with Multiple Copies of Atoms in the Ensemble

4 Neural-Symbolic Abductive Logic Programming

INTRO TO ALP

In addition to explaining the goal, abductive explanations are often required to satisfy some additional preference criteria. It is usual to restrict abductive explanations to some subset of ground atoms declared in advance as *abducibles.* Intuitively, abducibles are atoms whose truth is not initially known, but for which there may be partial information or integrity constraints restricting the combinations of abducibles that may appear in any explanation. Minimality requirements may also be imposed to ensure that abductive explanations contain no redundant atoms. These restrictions are elegantly and modularly expressed in the formalism of Abductive Logic Programming (ALP) [Kakas et al., 1992], which is an extension of standard logic programming for representing and reasoning about uncertainty.

Formally, an abductive logic program consists of a theory T, a set of abducibles A and a set of integrity constraints IC of the form $a_1, \ldots, a_n \rightarrow \perp$. Here, $\perp$ is the atom denoting falsity and the above constraint states that the atoms $a_1, \ldots, a_n$ must not all be true at the same time. Given a goal G, the task of ALP is to find an explanation $\Delta \subseteq A$ such that $T \cup \Delta$ entails G and satisfies IC. In addition, an explanation Δ is said to be *minimal* if there is no strict subset $\Delta' \subset \Delta$ such that Δ' is itself an explanation of G with respect to T.

Example 4.1 *Consider the abductive problem in Figure 4 below which concerns an old car. The theory states that the car won't start if its battery is flat or its fuel tank is empty, that the battery is flat on wet days, that the car will overheat if its fan is broken, and that the lights of the car are on. The integrity constraint states that the lights cannot be on at the same time that the battery is flat. The goal is to find out why the car won't start (wont_start). The possible explanations to be considered (abducibles) are wet_day, fan_broke and fuel_empty.*

$$
\begin{array}{lcl}
T & = & \left\{ \begin{array}{l} battery\_flat \rightarrow wont\_start \\ fuel\_empty \rightarrow wont\_start \\ wet\_day \rightarrow battery\_flat \\ fan\_broke \rightarrow overheat \\ lights\_on \end{array} \right\} \\
G & = & \{ \ wont\_start \ \} \\
IC & = & \{ \ battery\_flat, lights\_on \rightarrow \bot \ \} \\
A & = & \{ \ fan\_broke, fuel\_empty, wet\_day \ \}
\end{array}
$$

Fig. 4: An Abductive Logic Program

There are two correct explanations: $\Delta_1 = \{fuel\_empty\}$ *and* $\Delta_2 = \{fan\_broke, fuel\_empty\}$. *The former is minimal but the latter not (as it is a superset of the former). These are the only correct explanations since all other sets of abducibles fail to satisfy either the goal or the integrity constraints.*

Using the translation described in Section 2, the theory T can be represented by the neural network shown within the rounded rectangle in Figure 5.[3] However, the abductive approach introduced in the previous section is not suited to computing the explanations required in this case, since it does not take into account the ideas that explanations should only contain abducible atoms or must satisfy integrity constraints.

A more discerning abductive approach that takes into account the above additional restrictions is schematically illustrated in Figure 5. The intuition is essentially that of computing hypotheses by inspecting the network's outputs once it has settled down. If the network activates a *goal* neuron and does not activate an *integrity constraint* neuron then the activated abducibles con-

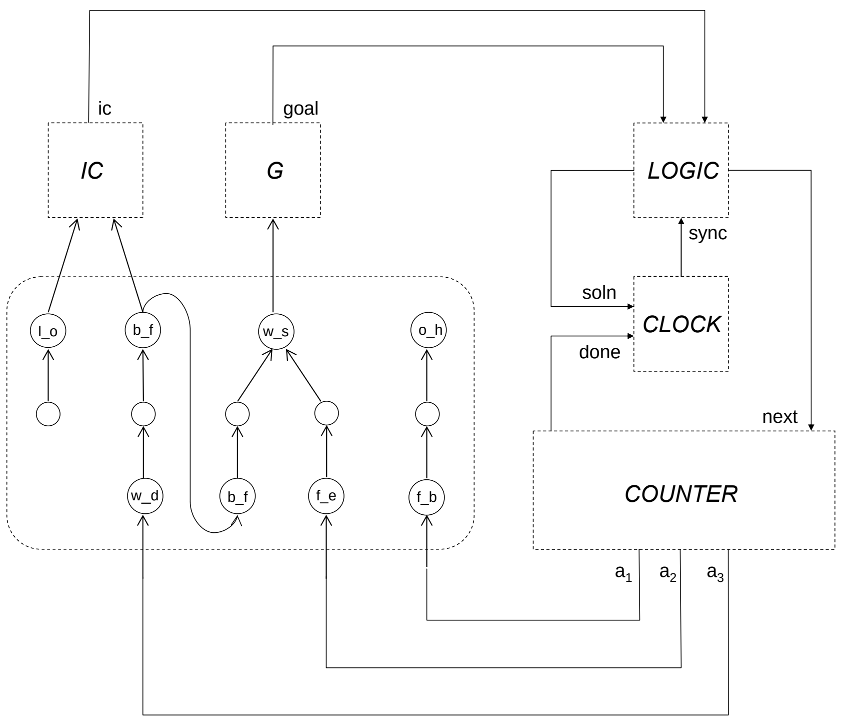

Fig. 5: A Neural-Symbolic System for an Abductive Logic Program

stitute a valid explanation. Conversely, if the network fails to activate the goal or activates an integrity constraint then the activated abducibles do not constitute a valid explanation and an alternative explanation must be sought.

This approach can be implemented as shown in Figure 5 using a binary counter to drive the abducibles. When the signal *next* is activated, the output of the counter, which is represented by the signals $a_1a_2a_3$, will advance to the next binary number 000, 001, 010, 011, etc. As the output changes, it activates a different combination of abducibles and causes the network corresponding to the theory to settle into a new state. The signal *goal* is true whenever all of the goals are satisfied, while the signal *ic* is true whenever some of the integrity constraints are violated. This information is fed to a simple control logic whose job is to route a periodic clock signal *sync* left or right according to whether the current abducibles comprise a valid explanation or not.

If the explanation is valid then the *sync* pulse is routed to the left, where it activates the *soln* signal and suspends the activity of the clock. In this case the entire system enters a stable state in which *soln* is active, indicating that an abductive explanation has been successfully found. If the explanation is invalid then the *sync* pulse is routed to the right, where it activates the *next* signal, thereby causing the counter to advance and the next set of abducibles to be considered. Once all possible solutions have been tried (i.e. the counter reaches 111, in this case) the counter overflows and activates the signal *done* which suspends the operation of the clock and indicates that there are no further hypotheses to investigate.

As explained in [Ray and dAvila Garcez, 2006], all of the components shown in Figure 5 can be implemented by expressing them as logic programs and translating them into neural networks in the usual way. For example, the *goal* signal is obtained from G by adding the following rule to the theory T:

$$wont\_start \rightarrow goal.$$

Similarly, the *ic* signal is obtained from IC by adding the following rule to T:

$$battery\_flat, lights\_on \rightarrow ic.$$

The abducibles A are connected to the output of the counter by adding three rules:

$$\left\{ \begin{array}{l} a_1 \rightarrow fan\_broke \\ a_2 \rightarrow fuel\_empty \\ a_3 \rightarrow wet\_day \end{array} \right\}$$

The *counter* itself is represented by the theory:

$$\bigcup_{i=1}^{N} \left\{ \begin{array}{l} a_i, \neg c_i \rightarrow a_i \\ d_i \rightarrow a_i \\ a_i \rightarrow b_i \\ b_{i-1}, \neg a_{i-1}, a_i \rightarrow c_i \\ b_{i-1}, \neg a_{i-1}, \neg a_i \rightarrow d_i \end{array} \right\} \cup \left\{ \begin{array}{l} next \rightarrow b_0 \\ b_N, \neg a_N \rightarrow done \\ done \rightarrow done \end{array} \right\}$$

where N is the number of abducibles and $\neg$ denotes negation.[4] When this theory is translated into a neural network, each bit of the counter uses four neurons, a_i, b_i, c_i and d_i, to implement a divide by two register that toggles the state of a_i whenever the state of a_{i-1} changes from *on* to *off* – with the neurons c_i and d_i signalling a_i to turn *off* and *on*, respectively.[5]

The *clock* is represented by the theory:

$$\bigcup_{i=1}^{M} \left\{ \; k_{i-1} \rightarrow k_i \; \right\} \cup \left\{ \begin{array}{l} \neg hold, \neg k_M \rightarrow k_0 \\ k_0, \neg k_1 \rightarrow sync \end{array} \right\}$$

where M is a constant that determines the period of the clock. Intuitively the neurons k_i are connected in a loop such that the state of each neuron follows that of its predecessor; except for the first, which always opposes the last. As mentioned above, the clock is disabled when *hold* is active. The output *sync* is active when k_0 is *on* but k_1 is not. The period of the clock is proportional to the number of neurons $M+1$, which must be chosen to give the rest of the network sufficient time to stabilise before the control logic decides whether the current explanation is valid or not.

The *control logic* is represented by the theory:

$$\left\{ \begin{array}{l} ic \rightarrow nogood \\ \neg goal \rightarrow nogood \\ sync, \neg nogood \rightarrow soln \\ soln, \neg nogood \rightarrow soln \\ soln \rightarrow hold \\ done \rightarrow hold \\ sync, nogood \rightarrow next \end{array} \right\}$$

where the atom *nogood* is true whenever the goal is not satisfied or one of the integrity constraints is violated. When *sync* becomes active, either *next* or *soln* will be activated depending on the state of *nogood*. The first case will advance the network into the next state while the second will force the network to stabilise in a solution state.

Assuming that all of its neurons are initially inactive, the network will quickly settle down into a stable state in which exactly one abducible,

$fuel\_empty$, is activated. The state corresponds to the abductive solution $\Delta = \{fuel\_empty\}$. If a signal is then applied to $next$, the network will leave this stable state and converge instead to another stable state representing the solution $\Delta = \{fan\_broke, fuel\_empty\}$. If another signal is applied to $next$ the network will leave this state and converge to a final state in which the neuron *done* is activated. This indicates that there are no more abductive solutions for this goal.

Consequently, this method provides a way of solving abductive logic programs using massively parallel neural hardware. If required, the approach can be made to enumerate all abductive solutions or provide an explicit guarantee when none exist. However, as it stands, there are two weaknesses which need to be addressed. First of all, a small modification is required to correctly handle *cyclic* logic programs which contain loops of the form $r_1 : a_1, \ldots \rightarrow a_2;\ r_2 : a_2, \ldots \rightarrow a_3; \ldots;\ r_n : a_n, \ldots \rightarrow a_1$ whereby an atom can potentially depend upon itself.

To see why cyclic programs are potentially problematic, suppose that the rule $fan\_broke \rightarrow over\_heat$ is added to T in the example above together wit the integrity constraint $\perp \rightarrow over\_heat$. The problem is that the cycle between fan_broke and $over\_heat$ introduces a memory into the network that causes these atoms to remain forever true after either one of them is activated – leading to a permanent violation of integrity thereafter. A solution to this problem is to add a new atom *true* into the body of each rule in the program and add a clause $\neg next \rightarrow true$. In this way, any self-sustaining loops are systematically deactivated just before the next set of abducibles is presented to the network.

The second weakness has to do with efficiency. The problem is that although the parallelism of the network is exploited when checking each individual hypothesis, the number of hypotheses checked is exponential in the number of abducibles (the abductive logic programming problem is known to be NP-complete even in the case of diagnostic problem solving [Bylander et al., 1991]). However, the search for minimal abductive explanations can be speeded-up by activating abducibles in the order 000, 001, 010, 100, 011, 101, etc, where all of the numbers with n bits high precede all of the numbers with $n+1$ bits high, etc. This can achieved by adding one layer of logic to the output of the counter.

It may be possible to further improve efficiency by utilising some form of pruning during the search as in symbolic ALP systems such as [Ray and Kakas, 2006], or employing some form of simplification when transforming the program as in the approach of *Answer Set Programming* [Simons et al., 2002]. However, this is still an open problem.

5 Conclusions and Future Work

We have presented two approaches to dealing with abduction in neural-symbolic systems. One approach makes few assumptions about abducibles or integrity constraints and uses CML to compute explanations in a top-down fashion. The other approach takes advantage of typical abduction set-ups which use abducibles and integrity constraints, and performs a more systematic, bottom-up computation.

COMPARISON OF TWO APPROACHES ALSO CITING [Aiello et al., 1995] and [Wang et al., 2006]

One may argue that the key difference between the two approaches is a mere exchange of space for time complexity, where the CML-based approach is *space intensive* and the ALP-based approach is *time intensive*, having a simpler structure. We believe that structure is key to computation, and should be investigated, in particular, in connection with learning. We accept that the systems proposed here have been engineered to perform specific tasks of symbolic abduction. It remains to be verified whether either of the proposed structures is more amenable to learning than the other. Both approaches use standard neural networks so that the exploitation of Backpropagation-based connectionist learning should be straightforward, although much empirical work is yet required in this area.

One may further argue that the machinery presented in Section 4 could be somewhat at odds with the broader Peircean conception of abduction, or indeed question whether the abductive logic programming approach based on the use of integrity constraints would be realistic in key application areas such as natural language interpretation [Hobbs et al., 1993]. If we compare our machinery with the Peircean apparatus generated in, for example, The Reach of Abduction [Gabbay and Woods, 2005], we see, among other things, that the former is a semantic treatment, whereas the latter is a pragmatic treatment. The semantic treatment has some advantages. One is that it is neat. Another is that it is the sort of treatment that readers will already have some familiarity with. Besides, since the semantic treatment can be seen as an abstraction from the pragmatic, we use it here mainly because, in virtually all respects, it allows us to lay down our proposals for abductive neural-symbolic integration. We accept that our system is a somewhat circumscribed instance of the Peircian or pragmatic model, and that in future work it should be investigated how our machinery adapts to the broader conception.

An alternative to the (localist) network structures proposed in this paper is the use of distributed representations as done in [Smolensky and Legendre, 2006]. As pointed out by one of the paper's referees, a major challenge for abduction is to account for creativity and the development of wholly new hypotheses [Perkins, 2000]. Distributed connectionist representations may be better equiped to deal with this problem than the logic-based approach, or indeed the localist connectionist approaches introduced here. This is a matter for investigation [Weber and Perkins, 1992]. As mentioned in the Introduction, we depart from distributed representations for two main reasons: localist representations can be associated with highly effective learning algorithms such as Backpropagation, and in our view localist networks are at an appropriate level of abstraction for symbolic knowledge represetation. As advocated in [Page, 2000], we believe one should be able to achieve the goals of distributed representations by properly changing the levels of abstraction of localist networks, while some of the desirable properties of localist models cannot be exhibited by fully distributed ones.

Of course, the question of how we humans perform abduction remains unanswered. But we argue that the prospects of answering this question are better if one investigates the connectionist processes of the brain in connection with the logical processes of symbolic computation, and not in isolation. The suggestion that abduction can be viewed as a connectionist logic is attractive for two reasons. One is that, unlike every other logic of explanation, connectionist explanation has a stab at being psychologically real (although once adapted to the broader, Peircian model, the claim to psychological reality may become more substantial than notional). The other, relatedly, is that a connectionist logic is no enemy of the subconscious and prelinguistic sectors of cognitive practice. But it is no panacea, either. As discussed above, the key question of how subsymbolic computation could solve the problem of the deployment of wholly new hypotheses, as, for example, in the case of Planck's postulation of quanta remains unanswered.

In summary, we believe that by paying attention to the developments on either side of the division between the symbolic and the non-symbolic approaches to abduction, we may be getting closer to a unifying theory, or at least promote a faster and principled development of the field. This paper describes and solves a very specific problem by two specific approaches. We hope it serves as a stepping stone in the modelling of abduction and in reconciling the symbolic and connectionist approaches. While it may have indeed posed more questions than answers, we hope it started to bridge this gap and contributed to placing the two research communities in closer contact.

Acknowledgements

We are grateful to Wilfrid Hodges, Johan vanBenthem, and the two anonymous referees for their valuable comments, which have helped improve this paper.

References

[Aiello et al., 1995] Aiello, A., Burattini, E., and Tamburrini, G. (1995). Purely neural, rule-based diagnostic systems ii: Uncertain reasoning. *International Journal of Intelligent Systems*, 10:751–769.

[Ayeb et al., 1998] Ayeb, B., Wang, S., and Ge, J. (1998). A Unified Model For Neural Based Abduction. *IEEE Transactions on Systems, Man and Cybernetics*, 28(4):408–425.

[Bylander et al., 1991] Bylander, T., Allemang, D., Tanner, M. C., and Josephson, J. R. (1991). The computational complexity of abduction. *Artificial Intelligence*, 49:25–60.

[Churchland, 1989] Churchland, P. (1989). *A Neurocomputational Perspective: The Nature of Mind and the Structure of Science.* MIT Press, Cambridge, MA.

[d'Avila Garcez et al., 2002] d'Avila Garcez, A. S., Broda, K., and Gabbay, D. (2002). *Neural-Symbolic Learning Systems: Foundations and Applications.* Perspectives in Neural Computing. Springer-Verlag.

[d'Avila Garcez et al., 2004] d'Avila Garcez, A. S., Lamb, L. C., Broda, K., and Gabbay, D. M. (2004). Applying connectionist modal logics to distributed knowledge representation problems. *International Journal of Artificial Intelligence Tools*, 13(1):115–139.

[d'Avila Garcez et al., 2006] d'Avila Garcez, A. S., Lamb, L. C., and Gabbay, D. M. (2006). Connectionist computations of intuitionistic reasoning. *Theoretical Computer Science*, 358(1):34–55.

[d'Avila Garcez et al., 2006] d'Avila Garcez, A. S., Lamb, L. C., and Gabbay, D. M. (2006). Connectionist modal logic. *Theoretical Computer Science*.

[Flach and Kakas, 2000] Flach, P. and Kakas, A., editors (2000). *Abduction and Induction: essays on their relation and integration*, volume 18 of *Applied Logic Series*. Kluwer.

[Gabbay and Woods, 2005] Gabbay, D. and Woods, J. (2005). *The reach of abduction: Insight and trial*. Elsevier.

[Goel and Ramanujam, 1996] Goel, A. and Ramanujam, J. (1996). A Neural Architecture for a Class of Abduction Problems. *IEEE Transactions on Systems, Man and Cybernetics*, 26(6):854–860.

[Hobbs et al., 1993] Hobbs, J. R., Stickel, M. E., Appelt, D. E., and Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence*, 63(1-2):69–142.

[Hughes and Cresswell, 1996] Hughes, G. E. and Cresswell, M. J. (1996). *A new introduction to modal logic.* Routledge, London and New York.

[Kakas et al., 1992] Kakas, A., Kowalski, R., and Toni, F. (1992). Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719–770.

[Kakas et al., 1994] Kakas, A. C., Kowalski, R. A., and Toni, F. (1994). The role of abduction in logic programming. In Gabbay, D. M., Hogger, C., and Robinson, J. A., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, pages 235–324. Oxford Science Publications.

[Levesque, 1989] Levesque, H. (1989). A knowledge-level account of abduction. In *International Joint Conference on Artificial Intelligence, IJCAI'89*, volume 2, pages 1061–1067.

[Lloyd, 1987] Lloyd, J. (1987). *Foundations of Logic Programming.* Springer Verlag.

[Michalski, 1993] Michalski, R. (1993). Inferential Theory of Learning: Developing Foundations for Multistrategy Learning. In Michalski, R. and Tecuci, G., editors, *Machine Learning: A Multistrategy Approach.* Morgan Kaufmann.

[Orgun and Ma, 1994] Orgun, M. A. and Ma, W. (1994). An overview of temporal and modal logic programming. In *Proc. Inl. Conf. Temporal Logic*, LNAI 827, pages 445–479. Springer.

[Page, 2000] Page, M. (2000). Connectionist modelling in psychology: A localist manifesto. *Behavioral and Brain Sciences*, 23:443–467.

[Perkins, 2000] Perkins, D. N. (2000). *The Eureka Effect: The Art and Science of Breakthrough Thinking.* Norton, New York.

[Ray, 2005] Ray, O. (2005). *Hybrid Abductive Inductive Learning.* PhD thesis, Department of Computing, Imperial College London.

[Ray and dAvila Garcez, 2006] Ray, O. and dAvila Garcez, A. (2006). Towards the integration of abduction and induction in artificial neural networks. In *Proceedings of the 2nd International Workshop on Hybrid Learning and Neural Networks.* to appear.

[Ray and Kakas, 2006] Ray, O. and Kakas, A. (2006). ProLogICA: a practical system for Abductive Logic Programming. In *Proceedings of the 11th International Workshop on Non-monotonic Reasoning.* to appear.

[Reggia et al., 1993] Reggia, J., Peng, Y., and Tuhrim, S. (1993). A Connectionist Approach to Diagnostic Problem-Solving Using Causal Networks. *Information Sciences*, 70:27–48.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press.

[Simons et al., 2002] Simons, P., Niemelä, I., and Soininen, T. (2002). Extending and implementing the stable model semantics. *Artificial Intelligece*, 138(1-2):181–234.

[Smolensky, 2000] Smolensky, P. (2000). Grammar-based connectionist approaches to language. *Cognitive Science*, 23:589–613.

[Smolensky and Legendre, 2006] Smolensky, P. and Legendre, G. (2006). *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*. MIT Press, Cambridge, Mass.

[Sun, 1995] Sun, R. (1995). Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence*, 75(2):241–296.

[Towell and Shavlik, 1994] Towell, G. G. and Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1):119–165.

[Wang et al., 2006] Wang, H., Johnson, T. R., and Zhang, J. (2006). A hybrid system of abductive tactical decision making. *International Journal of Hybrid Intelligent Systems*, 3(1):23–33.

[Weber and Perkins, 1992] Weber, R. J. and Perkins, D. N., editors (1992). *Inventive Minds: Creativity and Technology.* Oxford University Press, New York and Oxford.

[Zhang and Xu, 1999] Zhang, C. and Xu, Y. (1999). A Neural Network Model for Diagnostic Problem Solving with Causal Chaining. *Neural Networks and Advanced Control Strategies*, 54:87–92.

Notes

[1]For ease of reference, we normally prefix the rules by a label giving $r_i : a_1, ..., a_n \rightarrow a_0$. To avoid ambiguity, we use semi-colons to delineate the end of clauses.

[2]We are essentially labelling a, b and c here in order to be able to distinguish the two explanations. In the same way, we can now talk about a concept which holds in different networks (worlds) and distinguish, e.g., between $w_1 : a$ and $w_2 : a$.

[3] For convenience, predicate names have been abbreviated in the obvious way so that *battery_flat* is represented by *b_f* and so on.

[4]Here, negation is understood in the standard logic programming sense of *negation as failure* whereby the negative literal $\neg a$ is true if and only if the corresponding positive literal a is not true.

[5]It is interesting noting that in addition to its logical meaning, the $\rightarrow$ also has a temporal significance in these clauses. For example, the rule *done* $\rightarrow$ *done* is logically redundant, but when translated into a neural network, it ensures that if *done* is true in the *present* state of the network then it will also be true in the *next* state.

Artur S. d'Avila Garcez
Department of Computing
City University London
EC1V 0HB, London, UK
aag@soi.city.ac.uk

Dov M. Gabbay
Department of Computer Science
King's College London
WC2R 2LS, London, UK
dg@dcs.kcl.ac.uk

Oliver Ray
Department of Computing
Imperial College London
SW7 2BZ, London, UK
or@doc.ic.ac.uk

John Woods
Department of Philosophy
University of British Columbia
V6T 1Z4, Vancouver, Canada
JHWoods@interchange.ubc.ca