Cognitive Algorithms and Systems: Reasoning and Knowledge Representation

Article · December 2011

DOI: 10.1007/978-1-4419-1452-1_18

CITATIONS
6

READS
350

2 authors:

Artur D'Avila Garcez
City, University of London
160 PUBLICATIONS **1,837** CITATIONS

SEE PROFILE

Luís C. Lamb
Universidade Federal do Rio Grande do Sul
158 PUBLICATIONS **1,203** CITATIONS

SEE PROFILE

Chapter 18
Cognitive Algorithms and Systems: Reasoning and Knowledge Representation

Artur S. d'Avila Garcez and Luis C. Lamb

Abstract This chapter reviews recent advances in computational cognitive reasoning and their underlying algorithmic foundations. It summarises the neural-symbolic approach to cognition and computation. Neural-symbolic systems integrate two fundamental phenomena of intelligent behaviour: reasoning and the ability to learn from experience. The chapter illustrates how to represent, learn and compute several expressive forms of symbolic knowledge using neural networks. The goal is to provide computational models with integrated reasoning capabilities, where the neural networks offer the machinery for cognitive reasoning and learning while symbolic logic offers explanations to the neural models facilitating the necessary interaction with the world and other systems.

18.1 Introduction

Recent endeavors in cognitive science, artificial intelligence (AI) and evolutionary psychology have led to several hypotheses about the way cognitive models of reasoning, learning and language can be effected in or modelled by computational techniques (Pinker 2007; Pinker et al. 2008). Pinker has defended that the human mind is composed of computing constructions, or organs of computation (Pinker 2007). Furthermore, these models must cater for computation, specialisation and evolution. In computer science, recent efforts towards understanding and integrating learning, reasoning and action in artificial cognitive models have led to a number of developments, including approaches where learning and reasoning are modelled in a unified perspective (see, e.g. d'Avila Garcez et al. 2009; Valiant 2000). They have led also to the development of computational systems that are provably sound and have shown promise in a number of applications, including computational biology and fault diagnosis (d'Avila Garcez et al. 2002a).

A.S. d'Avila Garcez (✉)
Department of Computing, City University, London EC1V 0HB, UK
e-mail: aag@soi.city.ac.uk

V. Cutsuridis et al. (eds.), *Perception-Action Cycle: Models, Architectures, and Hardware*, Springer Series in Cognitive and Neural Systems 1,
DOI 10.1007/978-1-4419-1452-1_18,

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 (a notable exception is Mooney and Ourston 1994). For example, in AI, symbolic (logic-based) approaches have been mainly concerned with deductive reasoning, while connectionist (neural networks-based) approaches have mainly focused on inductive learning.

Neural-symbolic computation seeks to integrate the processes of logical reasoning and learning within the neural-computation paradigm (d'Avila Garcez 2005; d'Avila Garcez et al. 2007a, 2009; d'Avila Garcez and Lamb 2005). 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. 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. When we think of symbolic logic, we recognise its rigour, semantic clarity and the availability of automated proof methods which can provide explanations to the reasoning process, e.g. through a proof history. In neural computation, deduction can be seen as the network computation of output values as a response to input values, given a particular set of weights. Standard feedforward and partially recurrent networks have been shown capable of deductive reasoning of various kinds depending on the network architecture, including nonmonotonic (d'Avila Garcez et al. 2002a), modal (d'Avila Garcez et al. 2007b, 2009), intuitionistic (d'Avila Garcez et al. 2006a, b), epistemic (d'Avila Garcez and Lamb 2006; d'Avila Garcez and Lamb 2004) and abductive reasoning (d'Avila Garcez et al. 2007a).

In what follows, we briefly review the work on the integration of a range of computer science logics and neural networks. These constitute the technical foundations of a rich model of cognitive computation. In particular, we consider how standard neural networks can represent modal logic and its variations such as temporal logic. The resulting neural-symbolic cognitive system is called *connectionist modal logic* (CML). We then investigate how different networks and their associated logics can be combined to give an expressive yet feasible model of computation. For example, a network encoding some nonmonotonic mechanism of vision processing may need to be combined with a network that uses a temporal database for planning. A methodology for combining systems called the fibring method is used for this (d'Avila Garcez and Gabbay 2004). The overall model consists of an ensemble of simple single-hidden-layer neural networks – each may represent the knowledge of an agent (or a possible world) at a particular time-point – with connections between networks representing the relationships between agents/possible worlds. Each ensemble may be at a different level of abstraction, so that networks at one level may be fibred onto networks at another level to form a structure combining metalevel and object-level reasoning where high-level abstractions can be learned from

low-level concepts. We claim that this structure offers the basis for an expressive yet computationally tractable cognitive model for integrated robust learning and expressive reasoning.

18.2 Neurons and Symbols

The modelling of behaviour is an important goal of psychology, cognitive science, computer science, neural computation, philosophy, communication and other areas. Among the most prominent tools in the modelling of behaviour are computational-logic systems (e.g. classical logic, nonmonotonic logic, modal and temporal logic) and connectionist models of cognition (e.g. feedforward and recurrent networks, deep networks, self-organising networks).

The goal of neural-symbolic computation is to provide a coherent, unifying view for logic and connectionist network reasoning, contributing to the modelling and understanding of cognitive behaviour, and producing better computational tools. Typically, translation algorithms from a symbolic to a connectionist representation and vice versa are used to provide (a) a neural implementation of a logic, (b) a logical characterisation of a neural system, or (c) a hybrid learning system that brings together features from connectionism and symbolic AI.

In what follows, we focus on nonclassical logics and their associated recurrent-network models. In particular, we consider modal and temporal logics. Modal logics are among the most successful applied-logic systems (Blackburn et al. 2006). Temporal logic and its combination with other modalities such as knowledge operators have been the subject of intensive investigation leading to some of the main logical systems used in computer science and AI (Fagin et al. 1995; Vardi 1997). Recurrent networks, in turn, have been widely studied within neural computation and cognitive science, and applied to temporal sequence learning problems such as time-series prediction (Elman 1990). Our goal is to produce a robust computational system for modal and temporal knowledge representation using logic and connectionist recurrent networks. We claim that nonclassical reasoning has a major role to play in computer science. In addition, we subscribe to the view that computational cognitive modelling can lead to valid theories of cognition and offer a better understanding of certain cognitive processes (d'Avila Garcez et al. 2009; Sun 2009). Finally, we argue that a purely symbolic approach would not be sufficient, as also argued by Valiant (Valian 2008), and that a hybrid connectionist-symbolic approach can accommodate robustness and produce a more effective model of cognitive computation.

Our methodology is to transfer principles and mechanisms between nonclassical logic computation and neural computation. In particular, we consider how principles of symbolic computation can be implemented by connectionist mechanisms. The reason for this is that we see connectionism as the hardware to build upon with the use of different levels of abstraction according to the needs of the application. This methodology, looking at principles, mechanisms and applications, has proven a

fruitful way of progressing the research in the area (d'Avila Garcez et al. 2009) and abiding by Pinker's models of mind and cognition (Pinker 2007; Pinker et al. 2008). It has produced a connectionist system for nonclassical reasoning that strikes an adequate balance between complexity and expressiveness. In this system – known as a neural-symbolic system – neural networks provide the machinery for parallel computation and robust learning, while logic provides the necessary explanation to the network models, facilitating the necessary interaction with the world and other systems. In this integrated model, no conflict arises between a continuous and a discrete component of the system. Instead, a tightly coupled hybrid system exists that is continuous by nature (the neural network), but which has a clear discrete interpretation (its logic) at a different level of abstraction.

18.2.1 Abstraction

Growing attention has been given recently to deep network architectures where it is hoped that high-level abstract representations will emerge from low-level unprocessed datasets. The main example here are deep belief networks (Hinton et al. 2006), which use a sequence of restricted Boltzmann Machines to learn abstract representations from a grid of pixels, obtaining similar or better classification performance than support vector machines[1] (SVMs) (Shawe-Taylor and Cristianini 2004). This highlights the question of which representation is more appropriate in cognitive science: deep network models or shallow networks like SVMs. Neural-symbolic computation can help answer this question. It provides – almost as a side-effect – precise and useful expressiveness results for network models with respect to logic.

18.2.2 Modularity

Another key characteristic of neural-symbolic systems is modularity. In line with the ideas behind deep networks, neural-symbolic networks are modular and can be built through the careful engineering of network ensembles.[2] Modularity is of course important for comprehensibility and maintenance. On the other hand, massive integration of neural circuits is seen by many as a key feature of cognitive systems. There is a tension here between modularity and integration which emerges from

[1] It is worth noting that Hinton et al. (2006) is concerned mainly with unsupervised learning, while SVMs are supervised learning systems that require a certain amount of data preprocessing.

[2] Each network in the ensemble can be responsible for a specific task or logic, with the overall model being potentially very expressive. The methodology that we use to combine networks is that of fibring (Gabbay 1999) as discussed in some detail later.

computational concerns; it will probably fall on the field of cognitive computation to provide an alternative. Some strong hints as to the direction to follow can already be found in Taylor (2009).

18.2.3 Applications

Neural-symbolic systems have had important applications in many areas such as bioinformatics, simulation, robotics, fraud prevention and text processing. In such areas, a computational system is required to learn from experience and to reason about what has been learned (Browne and Sun 2001; Valiant 2003). For this process to be successful, the system must be robust (in the sense that the accumulation of errors resulting from the intrinsic uncertainty associated with the problem domain can be controlled). One such system that is already providing a contribution to problems in bioinformatics and engineering is the connectionist inductive learning and logic programming system (CILP) (d'Avila Garcez et al. 2002a). The merging of theory (known as background knowledge in machine learning) and data learning (i.e. learning from examples) in CILP networks have been shown more effective than purely symbolic or purely connectionist systems, especially in the case of noisy datasets (Towell and Shavlik 1994). Such results have contributed to the growing interest in developing neural-symbolic systems that are capable of learning from examples and background knowledge. It is important to consider the needs of the application. Complex applications can drive the research in this area further towards more effective systems.

18.2.4 Expressiveness

Until recently, neural-symbolic systems were not able to represent, compute and learn languages other than propositional logic and some fragments of first-order logic (Browne and Sun 2001; Cloete and Zurada 2000; Hölldobler and Kalinke 1994). In d'Avila Garcez et al. (2002b, 2003a, 2004c) and d'Avila Garcez and Lamb (2004), a new approach to knowledge representation and reasoning using neural-symbolic systems has been proposed, establishing a class of connectionist nonclassical logics, including connectionist modal, intuitionistic, temporal and epistemic logics (d'Avila Garcez et al. 2003b, 2004c; d'Avila Garcez and Lamb 2004). This new approach shows that a variety of nonclassical logics can be effectively represented by neural network ensembles. More recently, it has been shown that argumentation frameworks can also be represented by the same network ensemble models, offering an integrated approach to learning and reasoning of arguments, including non-standard forms of argumentation (d'Avila Garcez et al. 2004a, b, 2005) and of analogy (Borger et al. 2008).

As claimed in Browne and Sun (2001), if connectionism is a possible paradigm to cognitive science and AI, neural networks must be able to compute symbolic reasoning in an efficient and effective way. Moreover, in hybrid learning systems, usually the connectionist component is fault-tolerant, while the symbolic component may be "brittle and rigid." By integrating connectionist systems and sound nonclassical logics, we tackle this problem and offer a principled way to effectively compute, represent and learn various nonclassical logics within connectionist models.

18.2.5 Representation

A historical criticism of neural networks has been raised by John McCarthy already back in 1988 (McCarthy 1988). McCarthy referred to neural networks as having a "propositional fixation", in the sense that they were not able to represent first-order logic. This per se has remained a challenge for a decade, but several approaches have now dealt with first-order reasoning in neural networks, see, e.g. Browne and Sun (2001).

Perhaps in an attempt to address McCarthy's criticism, many researchers in the area have focused their attention only on first-order logic. More recently, it has been shown that nonclassical, practical reasoning can be used in a number of applications in neural-symbolic systems (d'Avila Garcez and Lamb 2004; d'Avila Garcez et al. 2003b, 2004a, b, c, 2005). Nonclassical logics have been shown adequate in expressing several reasoning features, allowing for the representation of temporal, epistemic and probabilistic abstractions in computer science and AI (Fagin et al. 1995; Gabbay et al. 2003; Halpern 2003). Some applications of nonclassical logics include the characterisation of timing analysis in combinatorial circuits (Mendler 2000) and in spatial reasoning (Bennett 1994), with possible use in geographical information systems. For instance, Bennett's propositional intuitionistic approach provided for tractable yet expressive reasoning about topological and spatial relations. Thus, a connectionist nonclassical logic can offer a richer cognitive model of computation, more realistic for modelling the many dimensions of an autonomous agent.

18.2.6 Nonclassical Reasoning

In summary, we believe that for neural computation to achieve its promise, connectionist models must be able to cater for nonclassical reasoning. We believe that the different communities cannot ignore the achievements and impact that nonclassical logics have had in computer science. Temporal logic, for instance, has had a large impact on both academia and industry (Gabbay et al. 1994; Pnueli 1977).

Modal logics, in turn, have become a lingua franca for the specification and analysis of knowledge and communication in multi-agent and distributed systems (Fagin et al. 1995; Wooldridge 2001). Epistemic logics have found a large number of applications, notably in game theory and in models of knowledge and interaction in multi-agent systems (Fagin et al. 1995; Gabbay et al. 1994; Pnueli 1977). Nonmonotonic reasoning has dominated the research on logic and AI in the 1980s and 1990s, and intuitionistic logic is considered by many as providing not only an adequate logical foundation for several core areas of theoretical computer science, including type theory and functional programming (van Dalen 2002), but also a solid basis for constructive reasoning (d'Avila Garcez et al. 2006a, b, 2009).

Notwithstanding all this evidence, little attention has been given to nonclassical reasoning and their integration with neural networks. If neural networks are to represent rich models of reasoning, it is undeniable that nonclassical logic should be at the core of this enterprise.

In the long run, neural-symbolic computation seeks to achieve a characterisation of a rich semantics for cognitive computation. This has been identified as a major challenge for computer science (Valiant 2003). We are proposing a methodology for the representation of several forms of nonclassical reasoning in artificial neural networks. Such expressive logics have been successfully used in computer science. Connectionist approaches should consider them by means of adequate computational models catering for integrated reasoning, knowledge representation and learning in cognitive science.

18.3 Neural-Symbolic Learning Systems

For neural-symbolic integration to be effective as a model of computation, we need to investigate how to represent, reason and learn expressive logics in neural networks. We also need to find effective ways of expressing the knowledge encoded in a trained network in a comprehensible symbolic form. There are at least two lines of action. The first is to take standard neural networks and try and find out which logics they can represent. The other is to take well-established logics and concepts (e.g. recursion) and try and encode those in a neural network architecture. Both lines require a principled approach, so that whenever we show that a particular logic can be represented by a particular neural network, we need to show that the network and the logic are in fact equivalent (a way of doing this is to prove that the network computes a formal semantics of the underlying logic). Similarly, if we develop a knowledge extraction algorithm, we need to make sure that it is correct (sound) in the sense that it produces rules that are encoded by the network, and that it is *quasi*-complete in the sense that the extracted rules increasingly approximate the exact behaviour of the network.

During the past 20 years, a number of models for neural-symbolic integration have been proposed [mainly in response to John McCarthy's note *Epistemological*

challenges for connectionism[3] (McCarthy 1988), itself a response to Paul Smolensky's *On the proper treatment of connectionism* (Smolensky 1988)]. Broadly speaking, researchers have made contributions in three main areas, providing (a) a logical characterisation of a connectionist system, (b) a connectionist implementation of a logic, or (c) a hybrid system bringing together features from connectionist systems and symbolic AI (Hitzler et al. 2004). Early relevant contributions include Hölldobler and Kalinke (1994), Shastri (1999) and Sun (1995) on knowledge representation, d'Avila Garcez and Zaverucha (1999) and Towell and Shavlik (1994) on learning with background knowledge, and Bologna (2004), d'Avila Garcez et al. (2001), Jacobsson (2005), Setiono (1997) and Thrun (1994) on knowledge extraction. The reader is referred to d'Avila Garcez et al. (2002a) for a detailed presentation of neural-symbolic learning systems and applications.

Neural-symbolic learning systems contain six main phases: (1) *background knowledge insertion,* (2) *inductive learning from examples,* (3) *massively parallel deduction,* (4) *theory fine-tuning,* (5) *symbolic knowledge extraction* and (6) *feedback* (see Fig. 18.1). In phase (1), symbolic knowledge is translated into the initial architecture of a neural network with the use of a *translation algorithm*. In phase (2), the neural network is trained with examples by a neural learning algorithm, which revises the theory given in phase (1) as *background knowledge*. In phase (3), the network can be used as a massively parallel system to compute the logical consequences of the theory encoded in it. In phase (4), information obtained from the computation carried out in phase (3) may be used to help fine-tune the network to better represent the problem domain. This mechanism can be used, for example, to resolve inconsistencies between the background knowledge and the training examples. In phase (5), the result of training is explained by the extraction of revised symbolic knowledge. As with the insertion of rules, the *extraction algorithm* must

[3] McCarthy (1988) identifies four knowledge representation problems for neural networks: the problem of *elaboration tolerance* (the ability of a representation to be elaborated to take additional phenomena into account), the *propositional fixation* of neural networks (based on the assumption that neural networks cannot represent relational knowledge), the problem of how to make use of any available *background knowledge* as part of learning and the problem of how to obtain domain *descriptions* from trained networks as opposed to mere discriminations. Neural-symbolic integration can address each of the above challenges. In a nutshell, the problem of elaboration tolerance can be resolved by having networks that are fibred forming a modular hierarchy, similar to the idea of using self-organising maps (Gärdenfors 2000; Haykin 1999) for language processing, where the lower levels of abstraction are used for the formation of concepts that are then used at the higher levels of the hierarchy. CML (d'Avila Garcez et al. 2007b) deals with the so-called propositional fixation of neural networks by allowing them to encode relational knowledge in the form of accessibility relations; a number of other formalisms have also tackled this issue as early as 1990 (Bader et al. 2005, 2007; Hölldobler 1993; Shastri and Ajjanagadde 1990), the key question being how to have simple representations that promote effective learning. Learning with background knowledge can be achieved by the usual translation of symbolic rules into neural networks. Problem descriptions can be obtained by rule extraction; a number of such translation and extraction algorithms have been proposed (e.g. Bologna 2004; d'Avila Garcez et al. 2001; d'Avila Garcez and Zaverucha 1999; Lozowski and Zurada 2000; Hitzler et al. 2004; Jacobsson 2005; Nunez et al. 2006; Setiono 1997; Sun 1995).

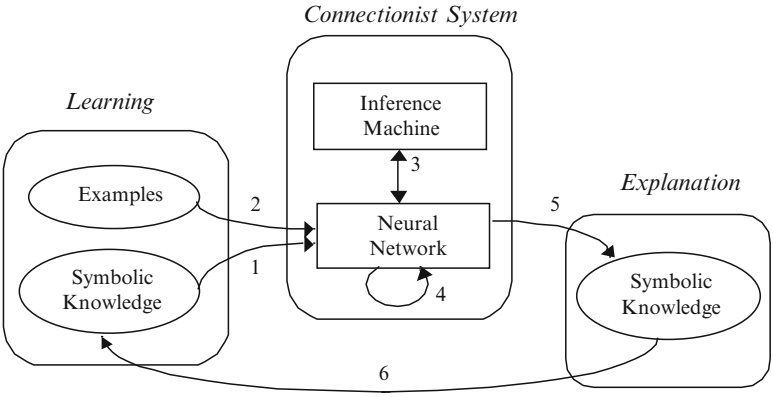

Fig. 18.1 Neural-symbolic learning systems

be provably correct, so that each rule extracted is guaranteed to be a rule of the network. Finally, in phase (6), the knowledge extracted may be analysed by an expert to decide whether it should feed the system again, closing the learning and reasoning cycle.

Our neural network models consist of feedforward and partially recurrent networks, as opposed to the symmetric networks investigated, e.g. in Smolensky and Legendre (2006). It uses a localist rather than a distributed representation,[4] and it works with backpropagation, the most successful neural learning algorithm used in industrial-strength applications (Rumelhart et al. 1986).

18.4 Technical Background

In this section, we introduce some technical aspects of neural-network and neural-symbolic computation. The reader can skip this section if interested mainly in the overall cognitive model architecture.

18.4.1 Neural Networks and Neural-Symbolic Systems

An artificial neural network is a directed graph. A unit (or neuron) in this graph is characterised, at time t, by its input vector $I_i(t)$, its input potential $U_i(t)$, its

[4] 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 representation. 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.

activation state $A_i(t)$ and its output $O_i(t)$. The units of the network are interconnected via a set of directed and weighted connections. If there is a connection from unit i to unit j, then $W_{ji} \in \mathbb{R}$ denotes the weight of this connection. The input potential of neuron i at time t ($U_i(t)$) 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 of a neuron i at time t ($A_i(t)$) is a bounded real or integer number. $A_i(t)$ is given by the neuron's *activation rule* h_i, which is a function of the neuron's input potential, i.e. $A_i(t) = h_i(U_i(t))$. Typically, h_i is either a linear, a non-linear or a sigmoid activation function, e.g. $tanh(x)$. In addition, θ_i is known as the threshold of neuron i. We say that neuron i is *activated* at time t if $A_i(t) > \theta_i$. Finally, the neuron's output value $O_i(t)$ is given by $f_i(A_i(t))$; usually, f_i is the identity function. The units of a neural network can be organised in layers. An n-layer feedforward network is an acyclic graph containing one input layer, $n-2$ hidden layers and one output layer. It computes a function $\varphi : \mathbb{R}^r \rightarrow \mathbb{R}^s$, where r and s denote the number of units occurring in the input and output layers, respectively. Most neural models also have a *learning rule*, responsible for changing the weights of the network so that it learns to approximate φ given a number of *training examples*, e.g. input vectors and their respective target output vectors.

The CILP (d'Avila Garcez et al. 2002a) is a computational model based on neural networks that integrates inductive learning from examples and background knowledge with deductive learning using logic programming. In CILP, a translation algorithm maps a logic program $\mathcal{P}$ into a single hidden layer neural network $\mathcal{N}$ such that $\mathcal{N}$ computes the fixed-point operator $\mathcal{T}_\mathcal{P}$ of $\mathcal{P}$. This provides a massively parallel model for computing the stable model semantics of $\mathcal{P}$. In addition, $\mathcal{N}$ can be trained with examples using a neural learning algorithm, having $\mathcal{P}$ as background knowledge. The knowledge acquired by training can then be extracted, closing the learning cycle (d'Avila Garcez et al. 2002a).

Let us exemplify how CILP translation algorithm works. Each rule (r_l) of $\mathcal{P}$ is mapped from the input layer to the output layer of $\mathcal{N}$ through one neuron (N_l) in the single hidden layer of $\mathcal{N}$. Intuitively, the translation algorithm from $\mathcal{P}$ to $\mathcal{N}$ has to implement the following conditions: (c_1). The input potential of a hidden neuron (N_l) can only exceed N_l's threshold (θ_l), activating N_l, when all the positive antecedents of r_l are assigned truth-value *true* while all the negative antecedents of r_l are assigned *false*; and (c_2). The input potential of an output neuron (A) can only exceed A's threshold (θ_A), activating A, when at least one hidden neuron N_l that is connected to A is activated.

Example 1. (CILP) Consider the logic program $\mathcal{P} = \{r_1 : B \wedge C \wedge \neg D \rightarrow A, r_2 : E \wedge F \rightarrow A, r_3 : B\}$, where $\neg$ stands for *default negation*. The translation algorithm derives the network $\mathcal{N}$ of Fig. 18.2, setting weights (W) and thresholds (θ) in such a way that conditions (c_1) and (c_2) above are satisfied. Note that if $\mathcal{N}$ ought to be fully connected, any other link (not shown in Fig. 18.2) should receive weight zero initially. Each input and output neuron of $\mathcal{N}$ is associated with an atom of $\mathcal{P}$. As a result, each input and output vector of $\mathcal{N}$ can be associated with an interpretation for $\mathcal{P}$, so that an atom (e.g. A) is true if its corresponding neuron (neuron A) is activated. Note also that each hidden neuron N_l corresponds

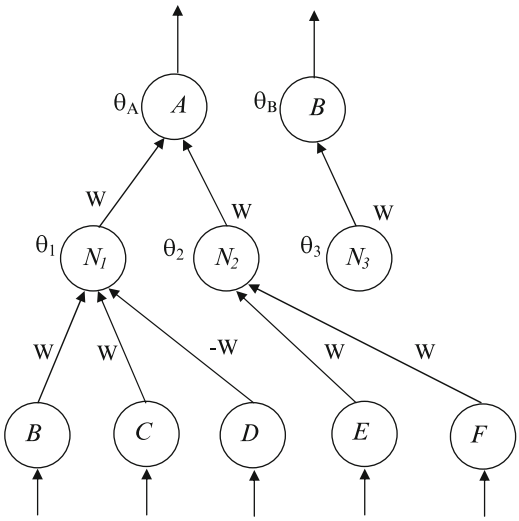

Fig. 18.2 Neural network for logic programming

to a rule r_l of $\mathcal{P}$. In order to compute a stable model, output neuron B should feed input neuron B such that $\mathcal{N}$ is used to iterate the fixed-point operator $\mathcal{T}_\mathcal{P}$ of $\mathcal{P}$ (d'Avila Garcez et al. 2002a). This is done by transforming $\mathcal{N}$ into a recurrent network $\mathcal{N}_\mathrm{r}$, containing feedback connections from the output to the input layer of $\mathcal{N}$, all with fixed weights $W_\mathrm{r} = 1$. In the case of $\mathcal{P}$ above, given any initial activation to the input layer of $\mathcal{N}_\mathrm{r}$, it always converges to the following stable state: $A = false$, $B = true$, $C = false$, $D = false$, $E = false$, and $F = false$, that represents the unique fixed-point of $\mathcal{P}$.

18.4.2 The Language of Connectionist Modal Logic

In CML, the CILP system is extended to the language of modal logic programming (Orgun and Ma 1994) further extended to allow modalities such as necessity ($\Box$) and possibility ($\Diamond$) to occur also in the head of clauses. The modalities shall allow us to represent several modes of reasoning, as we will illustrate in the coming sections. A modal translation algorithm then sets up an ensemble of CILP neural networks (d'Avila Garcez et al. 2002a), each network representing a possible world that can be trained by examples just like CILP networks. The ensemble computes a (fixed-point) semantics of modal theories, thus working as a massively parallel system for modal logic (d'Avila Garcez et al. 2004c). Since each network can be trained efficiently by a neural learning algorithm (e.g. backpropagation Rumelhart et al. 1986), one can adapt the ensemble by performing inductive learning.

A main feature of modal logics is the use of Kripke's *possible world semantics*. Under this interpretation, we say that 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. This is expressed in the semantics formalisation by a (binary) relation between possible worlds. In modal logic programming, a *modal atom* is of the form MA, where $M \in \{\Box, \Diamond\}$ and A is an atom. A *modal literal* is of the form ML, where L is a literal. A *modal program* is a finite set of clauses of the form $MA_1, \ldots, MA_n \rightarrow A$. We define *extended modal programs* as modal programs extended with modalities $\Box$ and $\Diamond$ also in the head of clauses and default negation in the body of clauses. In addition, each clause is labelled by the possible world in which it holds, similarly to Gabbay's labelled deductive systems (Broda et al. 2004). Thus, an *extended modal program* is a finite set of clauses C of the form $\omega_i : ML_1, \ldots, ML_n \rightarrow MA$, where ω_i is a label representing a world in which the associated clause holds, and a finite set of relations $\mathcal{R}(\omega_i, \omega_j)$ between worlds ω_i and ω_j in C.

A (Kripke) model M is a tuple $M = (\mathcal{W}, \mathcal{R}, \pi)$, where (a) $\mathcal{W}$ is a set of possible worlds; (b) $\mathcal{R}$ is a binary accessibility relation over worlds; and (c) π is a mapping associating worlds to formulas. We write $(M, \omega) \models \alpha$ if α is true at ω in M. Formally:

$(M, \omega) \models$ p iff $\omega \in \pi(p)$ for a propositional letter p
$(M, \omega) \models \neg\alpha$ iff $(M, \omega) \not\models \alpha$
$(M, \omega) \models \alpha \wedge \beta$ iff $(M, \omega) \models \alpha$ and $(M, \omega) \models \beta$
$(M, \omega) \models \Box\alpha$ iff $\forall\omega' \in \mathcal{W}$, if $\mathcal{R}(\omega, \omega')$ then $(M, \omega') \models \alpha$
$(M, \omega) \models \Diamond\alpha$ iff $\exists\omega'$ such that $\mathcal{R}(\omega, \omega')$ and $(M, \omega') \models \alpha$

When computing the semantics of a modal program, we consider what is computed in individual worlds, and the fixed-point of the program as a whole. When computing the fixed-point at each world, we consider the consequences derived locally and the consequences derived from the interaction between worlds. Locally, fixed-points are computed as before, by simply renaming each modal literal ML_i by a new literal L_j not in the language, and computing stable models. When considering interacting worlds, there are two cases to address, according to the $\Box$ and $\Diamond$ modalities and the accessibility relation $\mathcal{R}$, which might render additional consequences in each world.

Briefly, whenever $\Box A$ is true in a world (i.e. a neuron labelled $\Box A$ is activated in the corresponding neural network), A must be true in every world related to that (i.e. connections in the ensemble of networks must be established so that the firing of neuron $\Box A$ activates all neurons A in the related networks). Whenever $\Diamond A$ is true in a world (neuron $\Diamond A$ is activated), A must be true in one world related to that (connections must be established so that the firing of $\Diamond A$ activates A in one related world). The choice of the world in which to have A activated is arbitrary, reflecting the semantics of the $\Diamond$ modality. The following example illustrates this.

Example 2. (CML) Let $\mathcal{P} = \{\omega_1 : r \rightarrow \Box q, \omega_1 : \Diamond s \rightarrow r, \omega_2 : s, \omega_3 : q \rightarrow \Diamond p, \mathcal{R}(\omega_1, \omega_2), \mathcal{R}(\omega_1, \omega_3)\}$. We start by creating three CILP neural networks to represent the worlds ω_1, ω_2 and ω_3 (see Fig. 18.3). Then, we interconnect networks according to the meaning of $\Box$ and $\Diamond$. Hidden neurons labelled M, $\vee$ *and* $\wedge$ are

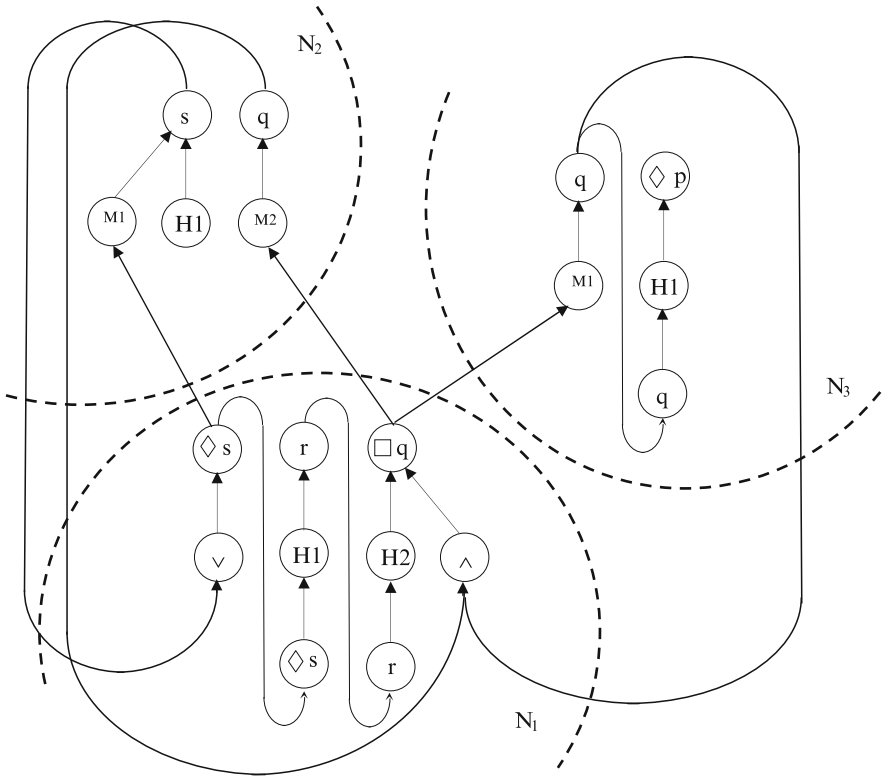

Fig. 18.3 Neural network ensemble for modal reasoning

created to do so. The remaining neurons are all created by CILP. For example, whenever neuron $\Box q$ is activated in ω_1, neuron q should be activated in both ω_2 and ω_3; whenever neuron $\Diamond s$ is activated in ω_1, neuron s should be activated in ω_2. This is implemented by using the hidden neurons labelled as M in the network. Dually, whenever q is activated in both ω_2 and ω_3, $\Box q$ should be activated in ω_1; whenever s is activated in ω_2, $\Diamond s$ should be activated in ω_1. This is implemented by using neurons labelled as $\wedge$ and $\vee$, respectively.

18.4.3 Reasoning About Time and Knowledge

In order to reason about the truth of sentences in time and represent knowledge evolution through time, we need to add temporal operators to the language of CML, as described below. We consider a temporal logic of knowledge that combines knowledge and time operators (see Fagin et al. 1995 for complete axiomatisations of such logics). The language of CML is extended with a set of agents $\mathcal{A} \subseteq \mathbb{N}$, a set of

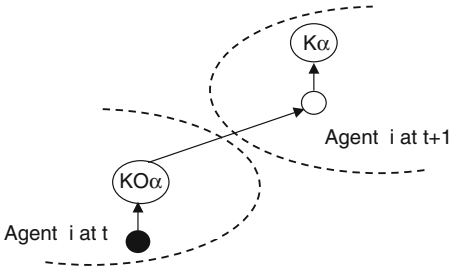

Fig. 18.4 Network ensemble for temporal reasoning

unary connectives: K_i, $i \in \mathcal{A}$, where $K_i p$ reads "agent i knows p", and the temporal operator $\bigcirc$ (next time). A temporal translation algorithm then is responsible for converting temporal rules of the form $t: K_{[\mathcal{A}]} L_1, ..., K_{[\mathcal{A}]} L_k \rightarrow \bigcirc K_{[\mathcal{A}]} L_{k+1}$, into neural network ensembles, where $[\mathcal{A}]$ denotes an element selected from $\mathcal{A}$ for each literal L_j, $1 \leq j \leq k+1$, $1 \leq t \leq n$; $k, n \in \mathbb{N}$ (d'Avila Garcez and Lamb 2004).

To each time-point, we associate the set of formulas holding at that point and extend the definition of a model M as follows: $M = (T, \mathcal{R}_1, ..., \mathcal{R}_n, \pi)$, where (a) T is a set of (linearly) ordered points; (b) $\mathcal{R}_i$ $(i \in A)$ is an agent accessibility relation over points; (c) π is a mapping associating time points to formulas. We write $(M, t) \models \alpha$ if α is true at point t in M. Formally:

$(M, t) \models \bigcirc \alpha$ iff $(M, t+1) \models \alpha$
$(M, t) \models K_i \alpha$ iff $\forall u \in T$, if $R_i(t, u)$ then $(M, u) \models \alpha$

It is worth noting that whenever a rule's consequent is preceded by $\bigcirc$, a forward connection from t to $t+1$ and a feedback connection from $t+1$ to t need to be added to the ensemble. For example, if $t: a \rightarrow \bigcirc b$ is a rule in $\mathcal{P}$, then not only must the activation of neuron a at t activate neuron b at $t+1$, but the activation of neuron b at $t+1$ must also activate neuron $\bigcirc b$ at t.

Example 3. One of the typical axioms of temporal logics of knowledge is $K_i \bigcirc \alpha \rightarrow \bigcirc K_i \alpha$ (Fagin et al. 1995), which means that an agent does not forget tomorrow what he knew today. This can be represented in an ensemble of CILP networks by connecting output neuron $K \bigcirc \alpha$ of agent i at time t to a hidden neuron that connects to output neuron $K\alpha$ of agent i at time $t+1$. In Fig. 18.4, the black circle denotes a neuron that is always activated (*true*), and the activation value of output neuron $K \bigcirc \alpha$ at time t propagates to output neuron $K\alpha$ at time $t+1$ via a hidden neuron. Weights must be such that $K\alpha$ at $t+1$ is also activated (*true*).

18.5 Connectionist Nonclassical Reasoning

As discussed in Sect. 18.2, we believe that for neural (and cognitive) computation to achieve their promise, their models must be able to cater for nonclassical reasoning. Nonclassical logics have had a great impact in philosophy and AI (d'Avila Garcez

and Lamb 2005; Fagin et al. 1995). For instance, nonmonotonic reasoning has dominated the research on logic and AI in the 1980s and 1990s, temporal logic has had a large impact on both academia and industry, and modal logics have become a lingua franca for the specification and analysis of knowledge and communication in multi-agent and distributed systems (Fagin et al. 1995). In this section, we consider modal and temporal reasoning as key representatives of nonclassical reasoning.

It is well known that modal logics correspond, in terms of expressive power, to the two-variable fragment of first-order logic (van Benthem 1984). Furthermore, as the two-variable fragment of first-order logic is decidable, this explains why modal logics are so "robustly decidable" and amenable to applications (Vardi 1997). Both AI and computer science have made extensive use of decidable modal logics, including in the analysis and model checking of distributed and multi-agent systems, program verification and specification, and hardware model checking. More recently, description logics, whose models are similar to modal logic (Kripke) models, have been instrumental in the study of the semantic web (Baader et al. 2003).

The basic idea behind connectionist nonclassical reasoning and CML is simple. Instead of having a single network as in the case of CILP, we now consider a set of CILP networks, and we label them, say, ω_1, ω_2, etc. Then, we can talk about a concept L holding at ω_1 and the same concept L holding at ω_2 separately. In this way, we can see ω_1 as a possible world and ω_2 as another, and this allows us to represent modalities such as necessity and possibility, time, arguments (d'Avila Garcez et al. 2005), epistemic states (d'Avila Garcez and Lamb 2006; d'Avila Garcez et al. 2003a, 2004c) and intuitionistic reasoning (d'Avila Garcez et al. 2006a, b, 2009). It is useful noting that this avenue of research is of interest in connection with McCarthy's conjecture on the propositional fixation of neural networks (McCarthy 1988), as discussed in Sect. 18.3, because of the correspondence between propositional modal logic and the above-mentioned two-variable fragment of first-order logic. In other words, CML shows that relatively simple neural-symbolic systems may go beyond classical propositional logic in terms of expressive power.

18.5.1 Connectionist Modal Reasoning

Modal logic deals with the analysis of concepts such as *necessity* (represented by $\Box L$, read "box L" and meaning that L is *necessarily true*), and *possibility* (represented by $\Diamond L$, read "diamond L" and meaning that L is *possibly true*). A key aspect of modal logic is the use of *possible worlds* and a binary (accessibility) relation $\mathcal{R}(\omega_i, \omega_j)$ between worlds ω_i and ω_j. In possible world semantics, 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.

CML uses ensembles of neural networks (instead of single networks) to represent the language of modal logic programming (Orgun and Ma 1994). The theories are now sets of modal clauses each of the form $\omega_i : ML_1, \ldots, ML_n \rightarrow MA$, where

ω_i is a label representing a world in which the associated clause holds and $M \in \{\Box, \Diamond\}$, together with a finite set of relations $\mathcal{R}(\omega_i, \omega_j)$ between worlds ω_i and ω_j. Such theories are implemented in a network ensemble, each network representing a possible world, with the use of labels in the ensembles allowing the representation of the accessibility relations.

In CML, each network in the ensemble is a simple single-hidden-layer CILP network 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 accessibility 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 neuron a must be activated in at least one of the 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. This is achieved by defining the connections and the weights of the network ensemble, following a translation algorithm.

Figure 18.3 is an example: it shows an ensemble of three neural networks labelled N_1, N_2 and N_3, which might *communicate* in different ways. We look at N_1, N_2 and N_3 as *possible worlds*. Input and output neurons may now represent $\Box L$, $\Diamond L$ or L, where L is a literal. $\Box A$ will be *true* in a world ω_i if A is *true* in all worlds ω_j to which ω_i is related. Similarly, $\Diamond A$ will be *true* in a world ω_i if A is *true* in some world ω_j to which ω_i is related. As a result, if neuron $\Box A$ is activated in network N_1, denoted by $\omega_1 : \Box A$, and world ω_1 is related to worlds ω_2 and ω_3, then neuron A must be activated in networks N_2 and N_3. Similarly, if neuron $\Diamond A$ is activated in N_1, then a neuron A must be activated in an arbitrary network that is related to N_1.

It is also possible to make use of CML to compute the fact that $\Box A$ holds at a possible world, say ω_i, whenever A holds at *all* possible worlds related to ω_i, by connecting the output neurons of the related networks to a hidden neuron in ω_i which connects to an output neuron labelled as $\Box A$. Dually for $\Diamond A$, whenever A holds at *some* possible world related to ω_i, we connect the output neuron representing A to a hidden neuron in ω_i which connects to an output neuron labelled as $\Diamond A$. Due to the simplicity of each network in the ensemble, when it comes to learning, we can still use backpropagation on each network to learn the local knowledge inside each possible world.

18.5.2 Connectionist Temporal Reasoning

The representation of temporal dimensions and symbolic temporal variables in cognitive science remains a relevant research field, with a number of implications not only in psychology but also in computer science and AI. Developments in this area demand, therefore, sound symbolic temporal inference systems underlying the neural and cognitive machinery. Existing approaches have now only started to make some headway towards sound representation of time in cognitive computational processes (d'Avila Garcez and Lamb 2006; Shastri 2007).

By extending the CML framework, we allow reasoning and learning about temporal, epistemic and probabilistic knowledge dealing with different reasoning dimensions of an idealised agent. Learning is achieved by training each individual network in the ensemble, which in turn corresponds to the current knowledge of an agent within a possible world. Such a form of learning aims to attend to the need of learning mechanisms in multi-agent systems in which modal logics are an essential feature to represent several kinds of knowledge dimensions an agent is typically endowed with (Wooldridge 2001).

The *Connectionist Temporal Logic of Knowledge* (CTLK) is an extension of CML, which considers temporal and epistemic knowledge (d'Avila Garcez and Lamb 2006). Generally speaking, the idea is to allow, instead of a single ensemble, a number n of ensembles, each representing the knowledge held by a number of agents at a given time-point t. Figure 18.5 illustrates how this dynamic feature can be combined with the symbolic features of the knowledge represented in each network, allowing not only for the analysis of the current state (possible world or time-point) but also for the analysis of how knowledge changes through time.

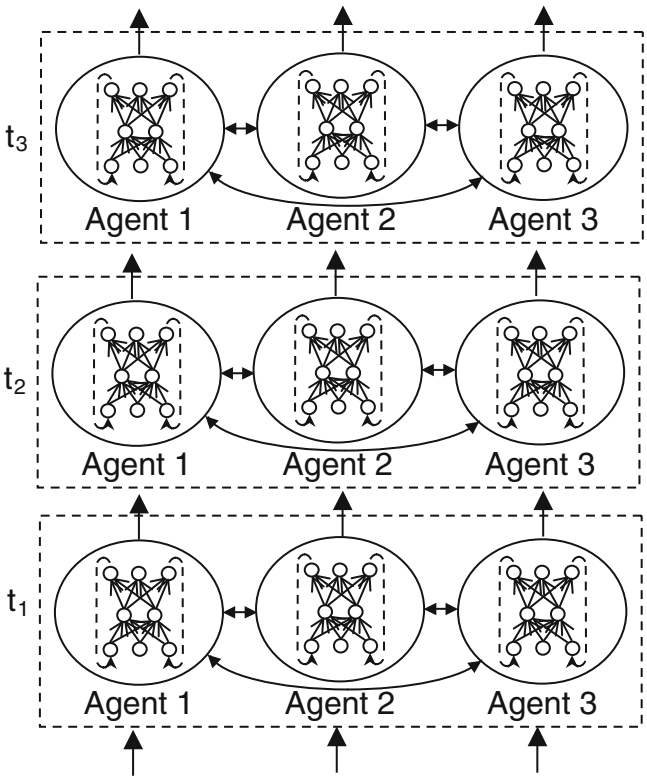

Fig. 18.5 Neural network ensemble for temporal reasoning

CML deals with time implicitly in snapshots; CTLK deals with time explicitly allowing the model to reason about time and knowledge. Different applications may be better suited to one or the other: the snapshot solution is simpler computationally, while the explicit model is richer. The *muddy children* case study, which we will consider in Sect. 18.5.3, will illustrate this.

The number of ensembles n that is necessary to solve a given problem will depend on the problem domain, in particular on the number of time-points needed for reasoning about the problem. For example, in the case of the *muddy children puzzle* (described below) (Fagin et al. 1995), which is a distributed knowledge representation problem, it is sufficient to use as many ensembles as the number of children that are muddy. The choice of n in a different domain might not be as straightforward, possibly requiring a fine-tuning process similar to that performed by learning, but with a varying network architecture. Other considerations around CTLK include the need for more extensive evaluations of the model with respect to learning, and the question of the trade-off between space and time complexity in such bounded networks. The fact, however, that the model is sufficient to deal with a variety of reasoning tasks is encouraging. Recently, CTLK was applied effectively on multi-process synchronisation and learning in concurrent programming (Lamb et al. 2007).

18.5.3 Case Study

In this section, we apply CTLK to the muddy children puzzle, a classic example of reasoning in multi-agent environments. In contrast to the also well-known wise men puzzle (Fagin et al. 1995; Huth and Ryan 2000), in which the reasoning process is sequential, in the muddy children puzzle, reasoning is distributed and simultaneous. There is a group of n children playing in a garden. A certain number of children k ($k \leq n$) has mud on their faces. Each child can see if the others are muddy, but cannot see if they themselves are muddy.[5]

A caretaker announces that at least one child is muddy ($k \geq 1$) and asks "do you know if you have mud on your face?" To help in the understanding of the puzzle, let us consider the cases where $k = 1$, $k = 2$ and $k = 3$.

If $k = 1$ (only one child is muddy), the muddy child answers *yes* at the first instance since she cannot see any other muddy child. All the other children answer *no* at the first instance.

If $k = 2$, suppose children 1 and 2 are muddy. In the first instance, all children can only answer *no*. This allows 1 to reason as follows: "if 2 had said *yes* the first time round, she would have been the only muddy child. Since 2 said *no*, she must be

[5] We follow the muddy children problem description presented in Fagin et al. (1995). We must also assume that all the agents involved in the situation are truthful and intelligent.

seeing someone else muddy, and since I cannot see anyone else muddy apart from 2, I myself must be muddy!" Child 2 can reason analogously and also answers *yes* at the second time round.

If $k = 3$, suppose children 1, 2 and 3 are muddy. Every child can only answer *no* at the first two time rounds. Again, this allows 1 to reason as follows: "if 2 or 3 had said *yes* at the second time round, they would have been the only two muddy children. Thus, there must be a third person with mud. Since I can see only 2 and 3 with mud, this third person must be me!" Children 2 and 3 can reason analogously to conclude as well that *yes*, they are muddy.

The puzzle illustrates the need to distinguish between an agent's individual knowledge and *common knowledge* about the world in each situation. For example, when $k = 2$, after everybody says *no* in the first round, it becomes common knowledge that at least two children are muddy. Similarly, when $k = 3$, after everybody says *no* twice, it becomes common knowledge that at least three children are muddy, and so on. In other words, when it is common knowledge that there are at least $k-1$ muddy children, after the announcement that nobody knows if they are muddy or not, then it becomes common knowledge that there are at least k muddy children, for if there were $k-1$ muddy children all of them would have known that they had mud on their faces. Note that this reasoning process can only start once it is common knowledge that at least one child is muddy, as announced by the caretaker.[6]

The snapshot version of the muddy children puzzle, where time is implicit, can be solved by CML. The interested reader is referred to d'Avila Garcez et al. (2009) for the details of the CML implementation. A full solution to the puzzle, however, can only be obtained with the use of CTLK. The addition of an explicit temporal variable to the puzzle allows one to reason about knowledge acquired after each time round. For example, assume as above that there are three muddy children playing in the garden. First, they all answer *no* when asked if they know whether they are muddy or not. Moreover, as each muddy child can see the other children, they will reason as previously described and answer *no* again at the second time round, reaching the correct conclusion at time round three. This solution requires, at each round, that the CML networks be extended with the knowledge acquired from reasoning about what is seen and what is heard by each agent. This clearly requires each agent to reason about time. There are alternative ways of modelling this, but one possible representation is as follows (below, a rule of the form $t_1 : \neg K_1 p_1 \wedge \neg K_2 p_2 \wedge \neg K_3 p_3 \rightarrow \bigcirc K_1 q_2$ states that if at time-point t_1, child 1 does not know that she is muddy, denoted by $\neg K_1 p_1$, and neither do children 2 and 3, then at the next

[6] The representation of common knowledge in neural networks throws some interesting questions. In CML, common knowledge is represented implictly by connecting neurons appropriately as reasoning progresses (e.g. as it becomes known at round two that at least two children should be muddy). The representation of common knowledge explicitly in the object level would require the use of neurons that are activated when "everybody knows" something (implementing in a finite domain the common knowledge axioms of Fagin et al. 1995), but this would complicate the formalisation of the puzzle given in this chapter. This explicit form of representation and its ramifications are worth investigating though and can be treated in their own right in future work.

time-point t_2, denoted by $\bigcirc$, child 1 knows that at least two children must be muddy, denoted by $K_1 q_2$):

Temporal rules for agent(child) 1:
$t_1 : \neg K_1 p_1 \wedge \neg K_2 p_2 \wedge \neg K_3 p_3 \rightarrow \bigcirc K_1 q_2$
$t_2 : \neg K_1 p_1 \wedge \neg K_2 p_2 \wedge \neg K_3 p_3 \rightarrow \bigcirc K_1 q_3$
Temporal rules for agent(child) 2:
$t_1 : \neg K_1 p_1 \wedge \neg K_2 p_2 \wedge \neg K_3 p_3 \rightarrow \bigcirc K_2 q_2$
$t_2 : \neg K_1 p_1 \wedge \neg K_2 p_2 \wedge \neg K_3 p_3 \rightarrow \bigcirc K_2 q_3$
Temporal rules for agent(child) 3:
$t_1 : \neg K_1 p_1 \wedge \neg K_2 p_2 \wedge \neg K_3 p_3 \rightarrow \bigcirc K_3 q_2$
$t_2 : \neg K_1 p_1 \wedge \neg K_2 p_2 \wedge \neg K_3 p_3 \rightarrow \bigcirc K_3 q_3$

The temporal rules above can be translated into a network structure similar to that of Fig. 18.5. Each network in the ensemble can be trained from examples using standard backpropagation. We have trained two groups of CTLK network ensembles to compute a solution to the muddy children puzzle. To one of them, we have added a temporal rule $t_1 : \neg K_1 p_1 \wedge \neg K_2 p_2 \wedge \neg K_3 p_3 \rightarrow \bigcirc K_1 q_2$ as background knowledge. To the other, we did not add any background knowledge. We then compared average test-set performances. We have considered the case in which Agent 1 is to decide whether or not she is muddy at time t_2. Each training example expresses the knowledge held by Agent 1 at t_2, according to the truth-values of atoms $K_1 \neg p_2$, $K_1 \neg p_3$, $K_1 q_1$, $K_1 q_2$, and $K_1 q_3$. As a result, there are 32 examples, i.e. all possible combinations of truth-values for input neurons $K_1 \neg p_2$, $K_1 \neg p_3$, $K_1 q_1$, $K_1 q_2$, $K_1 q_3$, with input value 1 denoting truth-value *true*, and input -1 denoting *false*. For each example, we are concerned about whether Agent 1 will know that she is muddy or not, i.e. whether output neuron $K_1 p_1$ is active.

From the description of the muddy children puzzle, we know that at t_2, $K_1 q_2$ should be *true* (i.e. $K_1 q_2$ is a *fact*). This information can be derived from the temporal rule given as background knowledge above, but not from the training examples. Although the background knowledge can be changed by the training examples, it places a bias on certain combinations (in this case, the examples in which $K_1 q_2$ is *true*), and this may produce a better performance, in particular when the background knowledge is correct. This effect has been observed, for instance, in Towell and Shavlik (1994) on experiments on DNA sequence analysis, in which background knowledge is expressed by *production rules*. In Towell and Shavlik (1994) (and also d'Avila Garcez et al. 2002a), the set of examples is noisy and the background knowledge counteracts the noise and reduces the chances of data overfitting.

We have evaluated the CTLK model using an eightfold *cross-validation* methodology, whereby eight CTLK network ensembles were created, each having been trained with 28 of the 32 available examples, with four examples being left out for testing at each time. This process was then repeated for the CTLK networks with background knowledge. For each training round, the training set was presented to the networks for 10,000 epochs. All the networks have reached a training-set error of 0.01. The ensembles containing no background knowledge achieved an average test-set accuracy of 81.25%. The ensembles to which the temporal rule was added as

background knowledge achieved an average test-set accuracy of 87.5%, a noticeable difference in performance. The result corroborates the importance of using background knowledge. In both cases, the same training parameters were used: learning rate $\eta = 0.2$, term of momentum $\alpha = 0.1$ and activation function $\tanh(x)$.

18.6 Fibring Neural Networks

In certain applications, different CTLK neural networks may need to be combined. In complex systems, it is frequently useful to create components that can be analysed independently and combined as appropriate in a modular way. A methodology for combining systems called *fibring* can be used for this (d'Avila Garcez and Gabbay 2004). Fibring promotes structured learning by combining networks at different levels of abstraction. Networks at one level may be fibred onto networks at another level to form a structure combining metalevel and object-level reasoning where high-level abstractions can be learned from low-level concepts.

In Bader et al. (2005), the idea of fibring was used to encode first-order logic programs in neural-network ensembles: a neural network is used to iterate a global counter and this counter is combined (fibred) with another neural network. This allows logic programs with an infinite number of ground instances to be translated into a finite neural network structure (e.g. $\neg even(x) \rightarrow even(s(x))$ for $x \in \mathbb{N}, s(x) = x + 1$). The translation is made possible because fibring implements a key feature of symbolic computation in neural networks, namely, *recursion*, as described below.

The idea of fibring neural networks is simple. Fibred networks may be composed not only of interconnected neurons but also of other networks, forming a recursive architecture. A fibring function then defines how this architecture behaves by defining how the networks in the ensemble should relate to each other. Typically, the fibring function will allow the activation of neurons in one network to influence the change of weights in another network. Intuitively, this can be seen as a master network being responsible for training a slave network. Interestingly, albeit being a combination of simple and standard neural networks, fibred networks are very expressive and can approximate any polynomial function in an unbounded domain, thus being more expressive than standard networks.

Figure 18.6 exemplifies how a network (B) can be fibred onto another network (A). Of course, the idea of fibring is not just to organise networks as a number of subnetworks; in Fig. 18.6, the output neuron of A is expected to be a neural network in its own right. The input, weights and output of B may depend on the activation state of A's output neuron, according to the fibring function φ. Fibred networks can be trained by examples in the same way as standard networks. For instance, networks A and B could have been trained separately before having been fibred. Networks can be fibred in a number of different ways as far as their architectures are concerned: network B could have been fibred onto a hidden neuron of network A.

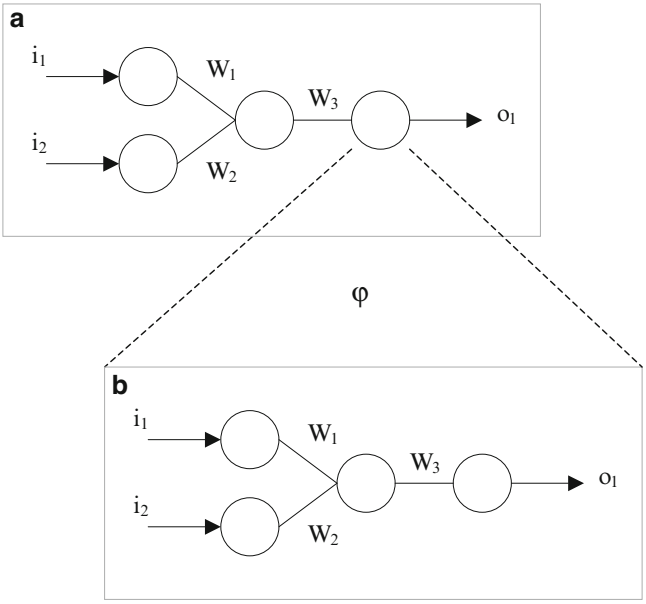

Fig. 18.6 Fibring neural networks

More generally, fibring offers a methodology for combining systems. As an example, network A could have been trained with a robot's visual system, while network B would have been trained with its planning system, and fibring would serve to perform the composition of the two systems (Gabbay 1999). Fibring can be very powerful. It offers the extra expressiveness required by complex applications at low computational cost (that of computing fibring function φ). Of course, we would like to keep φ as simple as possible so that it can be implemented itself by simple neurons in a fully connectionist model. Interesting work remains to be done in this area, particularly in what regards the question of emergence, modularity versus integration, and attentional focus (Taylor 2009), and the more practical question of how one should go about fibring networks in real applications. With respect to cognitive models, one can envisage fibring of several abilities: for instance, fibring of temporal and spatial reasoning with actions in an arbitrary environment can provide a useful framework to areas such as cognitive robotics.

18.7 Concluding Remarks

The need for rich, logic-based knowledge representation formalisms and algorithms in computational cognitive systems has been argued for a long time (d'Avila Garcez et al. 2009; Pinker 2007; Smolensky 1988; Stenning and van Lambalgen 2008; Valiant 1984). The foundational approach proposed in this chapter aims to attend

to such a need. The integration of other modes of reasoning, including conditional (Broda et al. 2002; Leitgeb 2007) and BDI logics (Rao and Georgeff 1998) would also contribute to the foundational model and corresponding experimental developments in neural-symbolic computation. The use of neural-symbolic systems also facilitates knowledge evolution and adaptability through learning. It would be interesting to apply the formalism to belief revision in the context of distributed, multi-agent systems.

CML and its variations offer an illustration of how the area of neural computation may contribute to the area of cognitive reasoning, while fibring is an example of how logic can bring insight into neural and cognitive computation. CML offers parallel models of computation for modal logics that can be integrated with an efficient learning algorithm. Fibring is an example of how concepts from symbolic computation, in this case *recursion*, can help in the development of new neural models. This is not necessarily conflicting with the ambition of biological plausibility, e.g. fibring functions can be understood as a model of *presynaptic weights*, which play an important role in biological neural networks.

Connectionist nonclassical reasoning and network fibring are the cornerstones of our overall cognitive model, which we may call *fibred network ensembles*. In this model, a network ensemble A (representing, e.g. a temporal theory) may be combined with another network ensemble B (representing, e.g. an intuitionistic theory d'Avila Garcez et al. 2006a). Higher level concepts (say, in A) may be combined and brought into the object-level (say, in B) without blurring the distinction between the two levels and maintaining modularity. One may reason in the metalevel and use that information in the object-level, a typical example being (metalevel) reasoning about actions in (object-level) databases containing inconsistencies (Gabbay and Hunter 1993). Relations between networks/concepts in the object-level may be represented and learned in the metalevel. If two networks denote, for example, concepts $P(X, Y)$ and $Q(Z)$, a meta-network can learn to map a representation of concepts P and Q onto a third network, denoting say $R(X, Y, Z)$, such that for example $P(X, Y) \wedge Q(Z) \rightarrow R(X, Y, Z)$. The interested reader is referred to d'Avila Garcez et al. (2009) for details.

Figure 18.7 illustrates the fibred network ensembles. The overall model takes the most general knowledge representation ensemble of Fig. 18.5 and allows a number of such ensembles to be combined at different levels of abstraction through fibring. Relations between concepts at level n may be generalised to level $n + 1$ with the use of metalevel networks. Abstract concepts at level n may be specialised (or instantiated) at level $n - 1$ with the use of a fibring function. Knowledge evolution through time occurs at each level. Alternative outcomes, possible worlds and the nonmonotonic reasoning process of multiple interacting agents can be modelled at each level. Learning can take place inside each modular network or across networks in the ensemble.

The question of how the human mind integrates reasoning and learning capacities is only starting to be answered (Gabbay and Woods 2005; Stenning and van Lambalgen 2008). We argue that the prospects are better if we investigate the connectionist processes of the brain together with the logical processes of symbolic

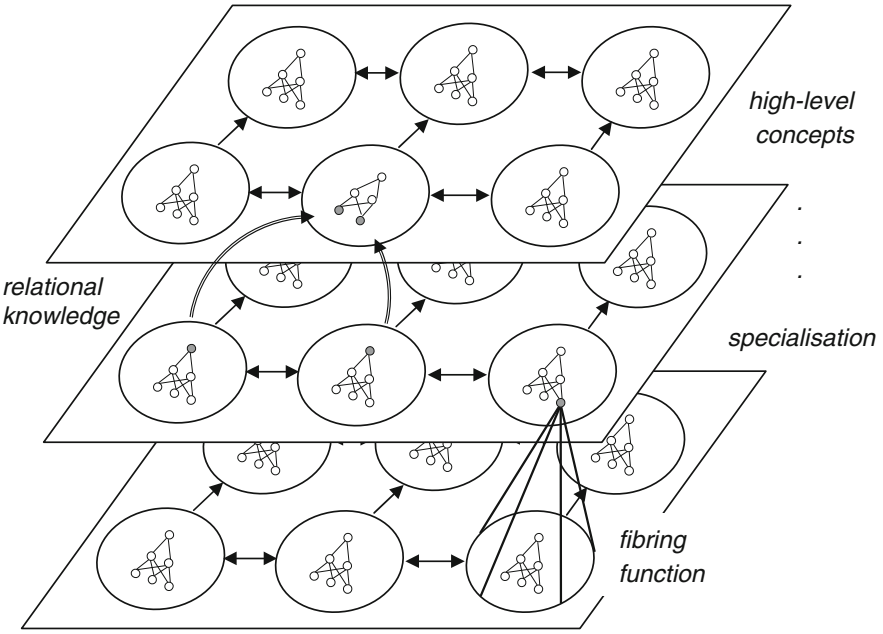

Fig. 18.7 Fibred network ensembles: structured learning and knowledge representation

computation, and not as two isolated paradigms. The framework of *fibred network ensembles* is expressive and tractable to address most current applications. Further development of the framework includes testing in controlled cognitive tasks.

The challenges for neural-symbolic cognitive computation today emerge from the goal of effective integration, expressive reasoning and robust learning. While adding reasoning capabilities to neural models, one cannot afford to lose learning performance. This means that one cannot depart from the key idea that neural networks are composed of simple processing units organised in a massively parallel architecture (i.e. one should not allow some clever neuron to perform complex symbol processing). It seems that learning is most effective at the propositional level, while some higher-level reasoning is useful at the first-order level. Computationally, the challenges are associated with the more practical aspects of the application of neural-symbolic cognitive systems in areas such as fault diagnosis, robotics, simulation and the semantic web. These challenges include the effective computation of logical models, the efficient extraction of comprehensible knowledge and, ultimately, the striking of the right balance between tractability and expressiveness. The reader is referred to d'Avila Garcez and Hitzler (2009) for a number of recent papers dealing with some of these challenges, including some real applications on multi-modal processing, simulation and defense.

In summary, by paying attention to the developments on either side of the division between the symbolic and the sub-symbolic paradigms, we are getting closer to a unifying theory of cognition, or at least promoting a faster and principled

development of the field of AI. This chapter described a family of connectionist nonclassical reasoning systems and hinted at how they may be combined at different levels of abstraction by fibring. We hope it serves not only as a stepping stone towards such a theory to reconcile the symbolic and connectionist approaches, but also as a foundational model for effective reasoning in cognitive computational systems.

Human beings are quite extraordinary at performing practical reasoning as they go about their daily business. *There are cases where the human computer, slow as it is, is faster than AI systems. Why are we faster? Is it the way we perceive knowledge as opposed to the way we represent it? Do we know immediately which rules to select and apply? We must look for the correct representation in the sense that it mirrors the way we perceive and apply the rules* (Gabbay 1998). Ultimately, neural-symbolic cognitive computation is about asking and trying to answer these questions, and about the associated provision of neural-symbolic systems with integrated expressive reasoning and robust learning capabilities.

References

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, Cambridge, 2003.

S. Bader, A. d'Avila Garcez, and P. Hitzler. Computing first-order logic programs by fibring artificial neural networks. In *Proceedings of the AAAI International FLAIRS Conference*, pages 314–319, 2005.

S. Bader, P. Hitzler, S. Holldobler, 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.

B. Bennett. Spatial reasoning with propositional logics. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning KR-94*, pages 51–62, 1994.

P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*. Studies in Logic and Practical Reasoning. Elsevier, Amsterdam, 2006.

G. Bologna. Is it worth generating rules from neural network ensembles? *Journal of Applied Logic*, 2(3):325–348, 2004.

R. V. Borger, A. d'Avila Garcez, and L. Lamb. A neural-symbolic perspective on analogy. *Behavioral and Brain Sciences*, 31(4):379–380, 2008.

K. Broda, D. Gabbay, L. Lamb, and A. Russo. Labelled natural deduction for conditional logics of normality. *Logic Journal of the IGPL*, 10(2):123–163, 2002.

K. Broda, D. Gabbay, L. Lamb, and A. Russo. *Compiled Labelled Deductive Systems: A Uniform Presentation of Non-classical Logics*. Studies in Logic and Computation. Research Studies Press/Institute of Physics Publishing, Baldock, UK, Philadelphia, PA, 2004.

A. Browne and R. Sun. Connectionist inference models. *Neural Networks*, 14:1331–1355, 2001.

I. Cloete and J. Zurada, editors. *Knowledge-Based Neurocomputing*. MIT, Cambridge, MA, 2000.

A. d'Avila Garcez. Fewer epistemological challenges for connectionism. In S. B. Cooper, B. Lowe, and L. Torenvliet, editors, *Proceedings of Computability in Europe, CiE 2005*, volume LNCS 3526, pages 139–149, Amsterdam, The Netherlands, June 2005. Springer, Berlin.

A. d'Avila Garcez and D. Gabbay. Fibring neural networks. In *Proceedings of 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 342–347, San Jose, CA, 2004.

A. d'Avila Garcez and P. Hitzler, editors. *Proceedings of IJCAI International Workshop on Neural-Symbolic Learning and Reasoning NeSy09*, Pasadena, California, USA, 2009.

A. d'Avila Garcez and L. Lamb. Reasoning about time and knowledge in neural-symbolic learning systems. In S. Thrun, L. Saul, and B. Schoelkopf, editors, *Advances in Neural Information Processing Systems 16*, Proceedings of NIPS 2003, pages 921–928. MIT, Cambridge, MA, 2004.

A. d'Avila Garcez and L. Lamb. Neural-symbolic systems and the case for non-classical reasoning. In S. Artëmov, H. Barringer, A. d'Avila Garcez, L. Lamb, and J. Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay*, pages 469–488. College Publications, International Federation for Computational Logic, UK, 2005.

A. d'Avila Garcez and L. Lamb. A connectionist computational model for epistemic and temporal reasoning. *Neural Computation*, 18(7):1711–1738, 2006.

A. d'Avila Garcez and G. Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intelligence Journal, Special Issue on Neural Networks and Structured Knowledge*, 11(1):59–77, 1999.

A. d'Avila Garcez, K. Broda, and D. Gabbay. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence*, 125:155–207, 2001.

A. d'Avila Garcez, K. Broda, and D. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. Perspectives in Neural Computing. Springer, Berlin, 2002a.

A. d'Avila Garcez, L. Lamb, and D. Gabbay. A connectionist inductive learning system for modal logic programming. In *Proceedings of the 9th International Conference on Neural Information Processing ICONIP'02*, pages 1992–1997, Singapore, 2002b. IEEE.

A. d'Avila Garcez, L. Lamb, K. Broda, and D. Gabbay. Distributed knowledge representation in neural-symbolic learning systems: A case study. In *Proceedings of AAAI International FLAIRS Conference*, pages 271–275, St. Augustine, FL, 2003a. AAAI.

A. d'Avila Garcez, L. Lamb, and D. Gabbay. Neural-symbolic intuitionistic reasoning. Frontiers in Artificial Intelligence and Applications Vol. 104, pages 399–408. IOS, 2003b.

A. d'Avila Garcez, D. Gabbay, and L. Lamb. Argumentation neural networks. In *Proceedings of the 11th International Conference on Neural Information Processing, ICONIP'04*, volume 3316 of *Lecture Notes in Computer Science*, pages 606–612. Springer, New York, 2004a.

A. d'Avila Garcez, D. Gabbay, and L. Lamb. Towards a connectionist argumentation framework. In *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI 2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22–27, 2004*, pages 987–988, 2004b.

A. d'Avila Garcez, L. Lamb, K. Broda, and D. Gabbay. Applying connectionist modal logics to distributed knowledge representation problems. *International Journal on Artificial Intelligence Tools*, 13(1):115–139, 2004c.

A. d'Avila Garcez, D. Gabbay, and L. Lamb. Value-based argumentation frameworks as neural-symbolic learning systems. *Journal of Logic and Computation*, 15(6):1041–1058, 2005.

A. d'Avila Garcez, L. Lamb, and D. Gabbay. Connectionist computations of intuitionistic reasoning. *Theoretical Computer Science*, 358(1):34–55, 2006a.

A. d'Avila Garcez, L. Lamb, and D. Gabbay. A connectionist model for constructive modal reasoning. In *Advances in Neural Information Processing Systems 18*, Proceedings of NIPS 2005, pages 403–410. MIT, 2006b.

A. d'Avila Garcez, D. M. Gabbay, O. Ray, and J. Woods. Abductive reasoning in neural-symbolic systems. *TOPOI: An International Review of Philosophy*, 26:37–49, 2007a.

A. d'Avila Garcez, L. Lamb, and D. Gabbay. Connectionist modal logic: Representing modalities in neural networks. *Theoretical Computer Science*, 371(1–2):34–53, 2007b.

A. d'Avila Garcez, L. Lamb, and D. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, Berlin, 2009.

J. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT, Cambridge, MA, 1995.

D. Gabbay. *Elementary Logics: a Procedural Perspective*. Prentice Hall, London, 1998.

D. Gabbay. *Fibring Logics*. Oxford University Press, Oxford, 1999. Oxford Logic Guides, Vol. 38.
D. M. Gabbay and A. Hunter. Making inconsistency respectable: Part 2 – meta-level handling of inconsistency. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty ECSQARU'93*, volume LNCS 747, pages 129–136. Springer, Berlin, 1993.
D. Gabbay and J. Woods. *A Practical Logic of Cognitive Systems, Volume 2: The reach of abduction: Insight and trial*. Elsevier, New York, 2005.
D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal logic: mathematical foundations and computational aspects*, volume 1. Oxford University Press, Oxford, 1994. Oxford Logic Guides, Vol. 28.
D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-dimensional Modal Logics: Theory and Applications*, volume 148 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science, Amsterdam, The Netherlands, 2003.
P. Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. MIT, Cambridge, MA, 2000.
J. Halpern. *Reasoning About Uncertainty*. MIT, Cambridge, MA, 2003.
S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 1999.
G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.
P. Hitzler, S. Holldobler, and A. K. Seda. Logic programs and connectionist networks. *Journal of Applied Logic*, 2(3):245–272, 2004. Special Issue on Neural-Symbolic Systems.
S. Hölldobler. Automated inferencing and connectionist models. Postdoctoral Thesis, Intellektik, Informatik, TH Darmstadt, 1993.
S. Hölldobler and Y. Kalinke. Toward 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.
M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, Cambridge, 2000.
H. Jacobsson. Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation*, 17(6):1223–1263, 2005.
L. Lamb, R. Borges, and A. d'Avila Garcez. A connectionist cognitive model for temporal synchronisation and learning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence AAAI 2007*, pages 827–832. AAAI, 2007.
H. Leitgeb. Neural network models of conditionals: an introduction. In X. Arrazola and J. M. Larrazabal et al., editors, *Proceedings of ILCLI International Workshop on Logic and Philosophy of Knowledge, Communication and Action*, pages 191–223, Bilbao, 2007.
A. Lozowski and J. Zurada. Extraction of linguistic rules from data via neural networks and fuzzy approximation. In I. Cloete and J. Zurada, editors, *Knowledge-Based Neurocomputing*, pages 403–417. MIT, Cambridge, 2000.
J. McCarthy. Epistemological challenges for connectionism. *Behavioral and Brain Sciences*, 11(1):44, 1988.
M. Mendler. Characterising combinatorial timing analysis in intuitionistic modal logic. *Logic Journal of the IGPL*, 8(6):821–852, 2000.
R. Mooney and D. Ourston. A multistrategy approach to theory refinement. In R. Michalski and G. Teccuci, editors, *Machine Learning: A Multistrategy Approach*, volume 4, pages 141–164. Morgan Kaufmann, San Mateo, CA, 1994.
H. Nunez, C. Angulo, and A. Catala. Rule based learning systems for support vector machines. *Neural Processing Letters*, 24(1):1–18, 2006.
M. Orgun and W. Ma. An overview of temporal and modal logic programming. In *Proceedings of the International Conference on Temporal Logic ICTL'94*, volume 827 of *Lecture Notes in Artificial Intelligence*, pages 445–479. Springer, Berlin, 1994.
M. Page. Connectionist modelling in psychology: A localist manifesto. *Behavioral and Brain Sciences*, 23:443–467, 2000.
S. Pinker. *The Stuff of Thought: Language as a Window into Human Nature*. Viking, New York, 2007.

S. Pinker, M. A. Nowak, and J. J. Lee. The logic of indirect speech. *Proceedings of the National Academy of Sciences USA*, 105(3):833–838, 2008.
A. Pnueli. The temporal logic of programs. In *Proceedings of 18th IEEE Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.
A. Rao and M. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–343, 1998.
D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT, Cambridge, 1986.
R. Setiono. Extracting rules from neural networks by pruning and hidden-unit splitting. *Neural Computation*, 9:205–225, 1997.
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.
L. Shastri. Shruti: A neurally motivated architecture for rapid, scalable inference. In B. Hammer and P. Hitzler, editors, *Perspectives of Neural-Symbolic Integration*, pages 183–203. Springer, Heidelberg, 2007.
L. Shastri and V. Ajjanagadde. From simple associations to semantic reasoning: A connectionist representation of rules, variables and dynamic binding. Technical report, University of Pennsylvania, 1990.
J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.
P. Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 44:1–74, 1988.
P. Smolensky and G. Legendre. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*. MIT, Cambridge, MA, 2006.
K. Stenning and M. van Lambalgen. Human reasoning and cognitive science, 2008.
R. Sun. Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence*, 75(2):241–296, 1995.
R. Sun. Theoretical status of computational cognitive modeling. *Cognitive Systems Research*, 10(2):124–140, 2009.
J. Taylor. Cognitive computation. *Cognitive Computation*, 1(1):4–16, 2009.
S. Thrun. Extracting provably correct rules from artificial neural networks. Technical report, Institut für Informatik, Universität Bonn, 1994.
G. Towell and J. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1):119–165, 1994.
L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
L. Valiant. A neuroidal architecture for cognitive computation. *Journal of the ACM*, 47(5):854–882, 2000.
L. Valiant. Three problems in computer science. *Journal of the ACM*, 50(1):96–99, 2003.
L. Valiant. Knowledge infusion: In pursuit of robustness in artificial intelligence. In *Proceedings of the 28th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 415–422, Bangalore, India, 2008.
J. van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, chapter II.4, pages 167–247. D. Reidel Publishing Company, Dordrecht, 1984.
D. van Dalen. Intuitionistic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 5. Kluwer, Dordrecht, 2nd edition, 2002.
M. Vardi. Why is modal logic so robustly decidable. In N. Immerman and P. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *Discrete Mathematics and Theoretical Computer Science*, pages 149–184. DIMACS, 1997.
M. Wooldridge. *Introduction to Multi-agent Systems*. Wiley, New York, 2001.