Chapter 13
Conclusions

This chapter reviews the neural-symbolic approach presented in this book and provides a summary of the overall neural-symbolic cognitive model. The book deals with how to represent, learn, and compute expressive forms of symbolic knowledge using neural networks. We believe this is the way forward towards the provision of an integrated system of expressive reasoning and robust learning. The provision of such a system, integrating the two most fundamental phenomena of intelligent cognitive behaviour, has been identified as a key challenge for computer science [255]. Our goal is to produce computational models with integrated reasoning and learning capability, in which neural networks provide the machinery necessary for cognitive computation and learning, while logic provides practical reasoning and explanation capabilities to the neural models, facilitating the necessary interaction with the outside world.

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 [186]). For example, in artificial intelligence, symbolic (logic-based) approaches have been concerned mainly with deductive reasoning, whereas connectionist (neural-network-based) approaches have focused mainly on inductive learning. It is well known that this connectionist/symbolic dichotomy in artificial intelligence reflects a distinction between brain and mind, but we argue this should not dissuade us from seeking a fruitful synthesis of these paradigms [56, 58, 71].

In our research programme, we are seeking to integrate the processes of reasoning and learning within the neural-computation paradigm. 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 data set (a set of examples), allowing useful generalisations over unseen examples. When we think of symbolic logic, we recognise its rigour, and semantic clarity, and the availability of automated proof methods which can provide explanations of the reasoning process, for example through a proof history. In neural computation, deduction can be seen as a 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 [66], modal [79], intuitionistic [77], and abductive [58] reasoning.

Our goal is to offer a unified framework for learning and reasoning that exploits the parallelism and robustness of connectionist architectures. To this end, we have chosen to work with standard neural networks, whose learning capabilities have been already demonstrated in significant practical applications, and to investigate how they can be enhanced with more advanced reasoning capabilities. The main insight of this book is that, in order to enable effective learning from examples and background knowledge, one should keep network structures as simple as possible, and try to find the best symbolic representation for them. We have done so by presenting translation algorithms from nonclassical logics to single-hidden-layer neural networks. It was essential to show equivalence between the symbolic representations and the neural networks, thus ensuring a sound translation of background knowledge into a connectionist representation. Such results have made our nonclassical neural-symbolic systems into cognitive massively parallel models of symbolic computation and learning. We call such systems, combining a connectionist learning component with a logical reasoning component, *neural-symbolic learning systems* [66].

In what follows, we briefly review the work on the integration of nonclassical logics and neural networks presented in the book (see also [33]). We then look at the combination of different neural networks and their associated logics by the use of the fibring method [67]. As seen in Chap. 9, the overall model consists of a set (an ensemble) of simple, single-hidden-layer neural networks – each of which may represent the knowledge of an agent (or a possible world) at a particular time point – and connections between networks representing the relationships between agents/possible worlds. Each ensemble may be seen as being 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 information. We claim that this structure offers the basis for an expressive yet computationally tractable model of integrated, robust learning and expressive reasoning.

13.1 Neural-Symbolic Learning Systems

For neural-symbolic integration to be effective in complex applications, we need to investigate how to represent, reason with, 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 to find out which logics they can represent. The other is to take well-established logics and concepts (e.g. recursion) and try to encode these in a neural-network architecture. Both 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 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 in 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 last twenty years, a number of models for neural-symbolic integration have been proposed (mainly in response to John McCarthy's note 'Epistemological challenges for connectionism'[1] [176], itself a response to Paul Smolensky's paper 'On the proper treatment of connectionism' [237]). Broadly speaking, researchers have made contributions in three main areas, providing either (i) a logical characterisation of a connectionist system, (ii) a connectionist implementation of a logic, or (iii) a hybrid system bringing together features from connectionist systems and symbolic artificial intelligence [132]. Early relevant contributions include [135,229,242] on knowledge representation, [80, 250] on learning with background knowledge, and [30, 65, 144, 227, 244] on knowledge extraction. The reader is referred to [66] 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) *fine-tuning of the theory,* (5) *symbolic knowledge extraction,* and (6) *feedback* (see Fig. 13.1). 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

[1] In [176], McCarthy identified 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 so as to form a modular hierarchy, similarly to the idea of using self-organising maps [112, 125] 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. Connectionist modal logic [79] 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 [11, 13, 134, 230], the key question being that of 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, for example [30,65,80,132,144,165,192,227,242].

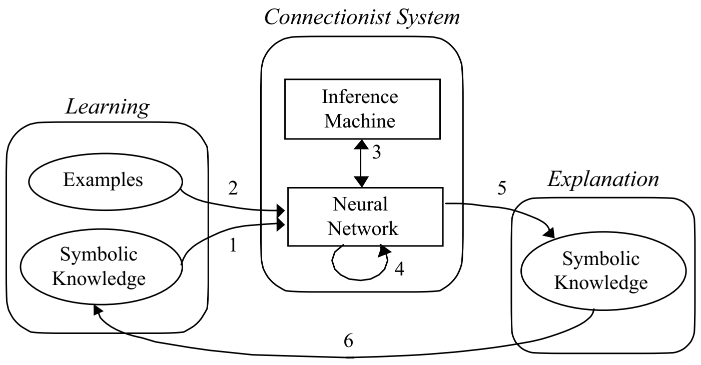

Fig. 13.1 Neural-symbolic learning systems

network in order 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 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 if 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, for example, in [240]. It uses a localist rather than a distributed representation,[2] and it works with backpropagation, the neural learning algorithm most successfully used in industrial-strength applications [224].

13.2 Connectionist Nonclassical Reasoning

We now turn to the question of expressiveness. We believe that for neural computation to achieve its promise, connectionist models must be able to cater for nonclassical reasoning. We believe that the neural-symbolic community cannot ignore the achievements and impact that nonclassical logics have had in computer science [71]. Whereas nonmonotonic reasoning dominated research in artificial intelligence in the 1980s and 1990s, temporal logic has had a large impact in both academia and industry, and modal logic has become a lingua franca for the specification and analysis

[2] 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 [200], we believe one should be able to achieve the goals of distributed representations by properly changing the levels of abstraction of localist networks, whereas some of the desirable properties of localist models cannot be exhibited by fully-distributed ones.

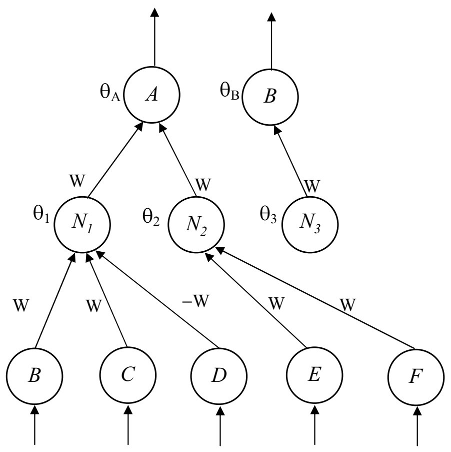

Fig. 13.2 Neural-network for logic programming

of knowledge and communication in multiagent and distributed systems [87]. In this section, we consider modal and temporal reasoning as key representatives of nonclassical reasoning.

The basic idea behind more expressive, connectionist nonclassical reasoning is simple, as seen in previous chapters of this book. Instead of having a single network, if we now consider a set of networks such as the one in Fig. 13.2, and we label them, say, as ω_1, ω_2, etc., then we can talk about a concept L holding in ω_1 and the same concept L holding in ω_2 separately. In this way, we can see ω_1 as one possible world and ω_2 as another, and this allows us to represent modalities such as necessity and possibility, and time, and also argumentation [69], epistemic states [74], and intuitionistic reasoning [77]. It is important to note that this avenue of research is of interest in connection with McCarthy's conjecture about the propositional fixation of neural networks [176] because there is a well-established translation between propositional modal logic and the two-variable fragment of first-order logic, as we have seen in Chap. 5 (see e.g. [264]), indicating that relatively simple neural-symbolic systems may go beyond propositional logic.

13.2.1 Connectionist Modal Reasoning

Recall that modal logic deals with the analysis of concepts such as *necessity* (represented by $\Box L$, read as 'box L', and meaning that L is *necessarily true*), and *possibility* (represented by $\Diamond L$, read as '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.

Connectionist modal logic (CML), presented in Chap. 5, uses ensembles of neural networks (instead of single networks) to represent the language of modal logic programming [197]. 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 ensembles and labels allowing the representation of accessibility relations.

In CML, each network in the ensemble is a simple, single-hidden-layer network like the network of Fig. 13.2, 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 information on how the networks should interact. For example, take three networks all related to each other. If a neuron $\Diamond a$ is activated in one of these networks, then a neuron a must be activated in at least one of the networks. If a 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 is achieved by defining the connections and the weights of the network ensemble, following a translation algorithm. Details of the translation algorithm, along with a soundness proof, have been given in Chap. 5.

As an example, consider Fig. 13.3. This shows an ensemble of three neural networks labelled N_1, N_2, 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

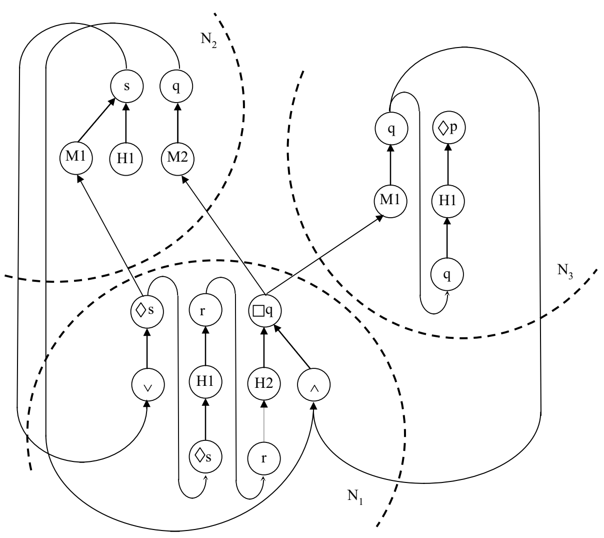

Fig. 13.3 Neural-network ensemble for modal reasoning

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 that $\Box A$ holds in a possible world, say ω_i, whenever A holds in *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 $\Box A$. Dually for $\Diamond A$, whenever A holds in *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 $\Diamond A$. Owing 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 in each possible world.

13.2.2 Connectionist Temporal Reasoning

The Connectionist Temporal Logic of Knowledge (CTLK), presented in Chap. 6, is an extension of CML which considers temporal and epistemic knowledge [72]. 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 13.4 illustrates how this dynamic feature can be combined with the symbolic features of the knowledge represented in each network, allowing not only the analysis of the current state (the current possible world or time point), but also the analysis of how knowledge changes over time.

Of course, there are important issues to do with (1) the optimisation of the model of Fig. 13.4 in practice, (2) the fact that the number of networks may be bounded, and (3) the trade-off between space and time computational complexity. The fact, however, that this model is sufficient to deal with such a variety of reasoning tasks is encouraging.

The definition of the number of ensembles s that are necessary to solve a given problem clearly depends on the problem domain, and on the number of time points that are relevant to reasoning about the problem. For example, in the case of the archetypal distributed-knowledge-representation problem of the muddy children puzzle [87], we know that it suffices to have s equal to the number of children that are muddy. The definition of s for a different domain might not be as straightforward, possibly requiring a fine-tuning process similar to that performed during learning, but with a varying network architecture. These and other considerations, including more extensive evaluations of the model with respect to learning, are still required. Recently, this model has been applied effectively to multiprocess synchronisation and learning in concurrent programming [157] and in a neural-symbolic model for analogical reasoning [34].

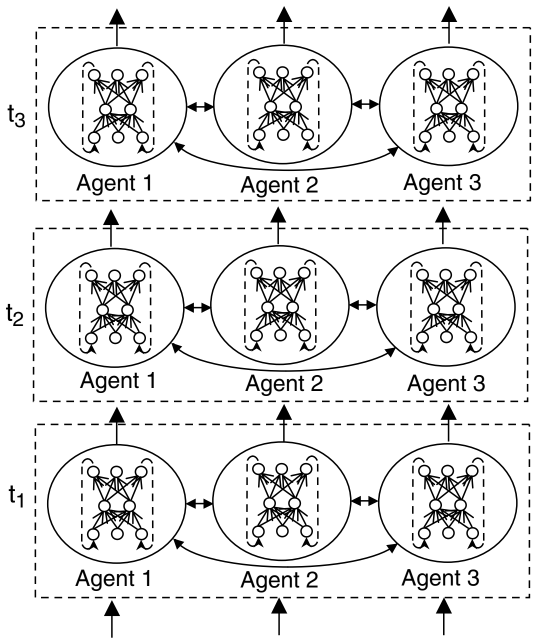

Fig. 13.4 Neural-network ensemble for temporal reasoning

13.3 Fibring Neural Networks

In CML, one may need to create copies of certain concepts. As a result, CML does not deal directly with infinite domains, since this would require infinitely many copies. An alternative is to map the instances of a variable onto the real numbers, and then use the real numbers as inputs to a neural network as a way of representing the instances. This has been done in [136]. However, the question of how to construct such a network to compute and learn a program remained unanswered, since no translation algorithm was given in [136]. A follow-up paper providing such an algorithm for first-order covered programs has appeared recently [13]. In [11], the idea of representing variables as real numbers was also followed, and a translation algorithm from first-order acyclic programs to neural-network ensembles was proposed. This algorithm makes use of *fibring* of neural networks [67]. Briefly, the idea is to use a neural network to iterate a global counter n. For each clause C_i in the logic program, this counter is combined (fibred) with another neural network, which determines whether C_i outputs an atom of level n for a given interpretation I. This allows us to translate programs with an infinite number of ground instances into a finite neural-network structure (e.g. $\neg even(x) \rightarrow even(s(x))$ for $x \in \mathbb{N}, s(x) = x+1$), and to prove that the network indeed approximates the fixed-point semantics of the program. The translation is made possible because fibring allows one to implement a key feature of symbolic computation in neural networks, namely *recursion*, as we describe 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 relate to each other. Typically, the fibring function will allow the activation of neurons in one network (A) to influence the change of weights in another network (B). Intuitively, this may be seen as training network B at the same time that network A runs. Interestingly, even though they are a combination of simple, standard neural networks, fibred networks can approximate any polynomial function in an unbounded domain, thus being more expressive than standard feedforward networks.

Figure 13.5, reproduced from Chap. 9, exemplifies how a network (B) can be fibred into another network (A). Of course, the idea of fibring is not only to organise networks as a number of subnetworks; in Fig. 13.5, the output neuron of A is expected to be a neural network (B) 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 that standard networks are. For instance, networks A and B could have been trained separately before they were fibred. Networks can be fibred in a number of different ways as far as their architectures are concerned; network B could have been fibred into a hidden neuron of network A.

As an example, network A could have been trained with a robot's visual system, while network B could have been trained with its planning system, and fibring would serve to perform the composition of the two systems [101]. Fibring can be very powerful. It offers the extra expressiveness required by complex applications at low computational cost (that of computing the fibring function φ). Of course, we would like to keep φ as simple as possible so that it can be implemented itself by

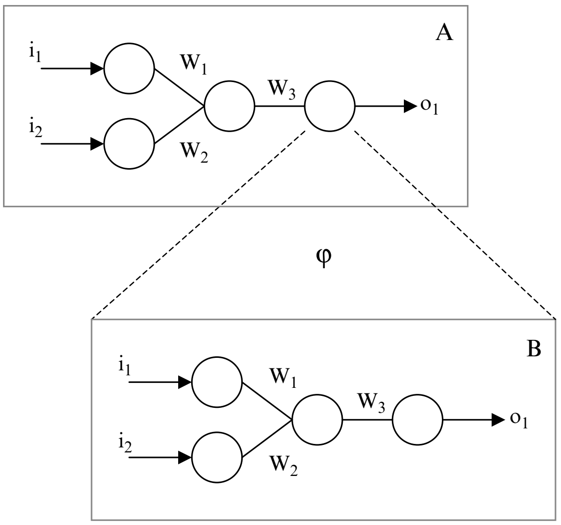

Fig. 13.5 Fibring neural networks

simple neurons in a fully-connectionist model. Interesting work remains to be done in this area, particularly in regard to the question of how one should go about fibring networks in real applications.

13.4 Concluding Remarks

Connectionist modal logic and its variations offer an illustration of how the area of neural networks may contribute to the area of logic, while fibring is an example of how logic can bring insight into neural computation. CML offers parallel models of computation to modal logic that, at the same time, can be integrated with an efficient learning algorithm. Fibring is a clear example of how concepts from symbolic computation may help in the development of neural models. This does not necessarily conflict with the ambition of biological plausibility; for example, 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 bring us to our overall cognitive model, which we may call *fibred network ensembles*. In this model, a network ensemble A (representing, for example, a temporal theory) may be combined with another network ensemble B (representing, for example, an intuitionistic theory). Along the same lines, metalevel concepts (in A) may be combined and brought into the object level (B), without necessarily blurring the distinction between the two levels. 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 [96]. Relations between networks/concepts in the object level may be represented and learned in the metalevel as described in Chap. 10. If two networks denote, for example, $P(X,Y)$ and $Q(Z)$, a metanetwork can learn to map a representation of the concepts P and Q (for instance using the hidden neurons of networks P and Q) onto a third network, denoting, say, $R(X,Y,Z)$, such that, for example, $P(X,Y) \wedge Q(Z) \rightarrow R(X,Y,Z)$.

Figure 13.6 illustrates fibred network ensembles. The overall model takes the most general knowledge representation ensemble of the kind shown in Fig. 13.4, and allows a number of such ensembles to be combined at different levels of abstraction through fibring. Relations between concepts at a level n may be generalised at 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. Evolution of knowlege with 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, simple network or across networks in the ensemble.

Both the symbolic and the connectionist paradigms have virtues and deficiencies. Research into the integration of the two has important implications that are beneficial to computing and to cognitive science. Concomitantly, the limits of effective integration should also be pursued, in the sense that integration might become

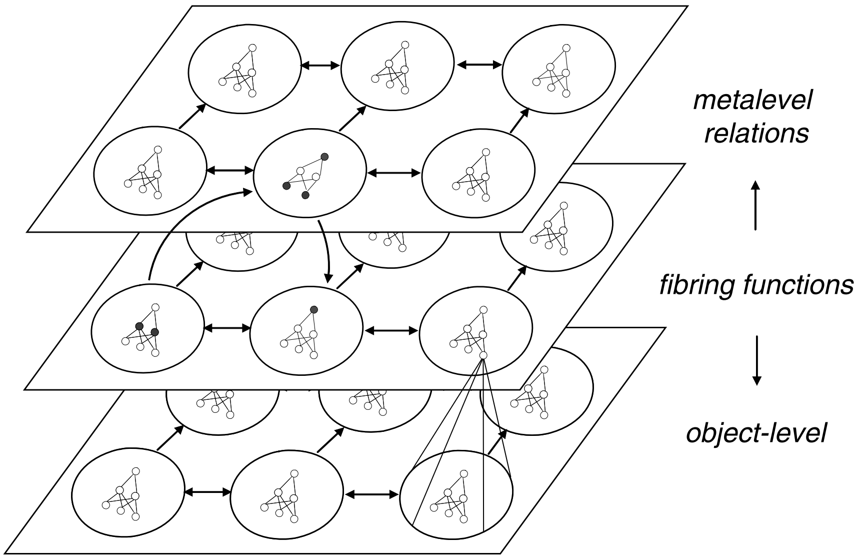

Fig. 13.6 Fibred network ensembles

disadvantageous in comparison with purely symbolic or purely connectionist systems. We believe that this book has contributed to the synergetic integration of connectionism and symbolism.

Of course, the question of how we humans integrate reasoning and learning is only starting to be answered [109, 241]. We argue that the prospects are better if we investigate the connectionist processes of the brain together with the logical processes of symbolic computation, rather than as two isolated paradigms. Of course, we shall need to be precise as we develop the framework further and test our model of fibred network ensembles in real cognitive tasks.

The challenges for neural-symbolic integration today emerge from the goal of effective integration of expressive reasoning and robust learning. One cannot afford to lose on learning performance when adding reasoning capability to neural models. This means that one cannot depart from the key concept that neural networks are composed of simple processing units organised in a massively parallel fashion (i.e. one cannot allow some clever neuron to perform complex symbol processing). Ideally, the models should be amenable to advances in computational neuroscience and brain imaging, which can offer data and also insight into new forms of representation. Finally, there are computational challenges associated with the more practical aspects of the application of neural-symbolic systems in areas such as engineering, robotics, and the Semantic Web. These challenges include the effective, massively parallel computation of logical models, the efficient extraction of comprehensible rules, and, ultimately, the striking of the right balance between computational tractability and expressiveness. References [57, 59–61] contain a number of recent papers dealing with some of today's challenges.

In summary, by paying attention to the developments on either side of the division between the symbolic and the subsymbolic paradigms, we are getting closer to a unifying theory, or at least promoting a faster, principled development of the cognitive and computing sciences and artificial intelligence. This book has 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 that it serves as a stepping stone towards such a theory to reconcile the symbolic and connectionist approaches.

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 artificial intelligence 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* [100]. Ultimately, neural-symbolic integration is about asking and trying to answer these questions, and about the associated provision of neural-symbolic systems with integrated capabilities for expressive reasoning and robust learning. We believe that this book has offered a principled computational model towards this goal.