1. Introduction and Overview

This book is about the integration of neural networks and symbolic rules. While symbolic artificial intelligence assumes that the *mind* is the focus of intelligence, and thus intelligent behaviour emerges from complex symbol processing mechanisms, connectionist artificial intelligence admits that intelligence lies in the *brain*, and therefore tries to model it by simulating its electrochemical neuronal structures. Clearly, such structures are capable of learning and performing the higher level cognitive tasks that human beings are accustomed to, as well as lower level, everyday cognitive activities. In this framework, the role of symbolic computation is to provide the system with the background information needed for the learning process, as well as to provide us with the information needed for understanding the system, since high-level cognitive tasks are much more clearly digested by human beings as symbols and operations over symbols rather than in the form of interconnected neurons.

1.1 Why Integrate Neurons and Symbols?

Human cognition successfully integrates the connectionist and symbolic paradigms of Artificial Intelligence (AI). Yet the modelling of cognition develops these separately in neural computation and symbolic logic/AI areas. There is now a movement towards a fruitful mid-point between these extremes, in which the study of logic is combined with connectionism. It is essential that these be integrated, thereby enabling a technology for building better intelligent systems.

The aim of *Neural-Symbolic Integration* is to explore and exploit the advantages that each paradigm presents. Among the advantages of artificial neural networks are massive parallelism, inductive learning and generalisation capabilities. On the other hand, symbolic systems can explain their inference process, e.g. through automated theorem proving, and use powerful declarative languages for *Knowledge Representation*.

In this book, we explore the synergies of neural-symbolic integration from the following perspective. We use a *Neural Network* to simulate a given task.

The network is obtained by being programmed (set up) and/or by somehow adapting and generalising over well-known situations (learning). The network is the mechanism to execute the task, while symbolic logic enables the necessary interaction between the network and the outside world.

"It is generally accepted that one of the main problems in building Expert Systems (which are responsible for the industrial success of Artificial Intelligence) lies in the process of knowledge acquisition, known as the *Knowledge Acquisition Bottleneck*" [LD94]. An alternative is the automation of this process using *Machine Learning* techniques (see [Mit97, Rus96]). Symbolic machine learning methods are usually more effective if they can exploit background knowledge (incomplete domain theory). In contrast, neural networks have been successfully applied as a learning method from examples only (data learning) [TBB+91, SMT91]. As a result, the integration of theory and data learning in neural networks seems to be a natural step towards more powerful training mechanisms.[TS94a]

The *Inductive Learning* task employed in symbolic machine learning is to find hypotheses that are consistent with a background knowledge to explain a given set of examples. In general, these hypotheses are definitions of concepts described in some logical language. The examples are descriptions of instances and non-instances of the concept to be learned, and the background knowledge provides additional information about the examples and the concepts' domain knowledge [LD94].

In contrast with (symbolic) learning systems, the learning of (numeric) neural networks implicitly encodes patterns and their generalisations in the networks' weights, so reflecting the statistical properties of the trained data [BL96]. It has been indicated that neural networks can outperform symbolic learning systems, especially when data are noisy[1] [TBB+91]. This result, due also to the massively parallel architecture of neural networks, contributed decisively to the growing interest in combining, and possibly integrating, neural and symbolic learning systems (see [Kur97] for a clarifying treatment of the suitability of neural networks for the representation of symbolic knowledge).

> "We believe that neural networks are capable of more than pattern recognition; they can also perform higher cognitive tasks which are fundamentally rule-governed. Further, we believe that they can perform higher cognitive tasks better if they incorporate rules rather than eliminate them. A number of well known cognitive models, particularly of language, have been criticised for going too far in eliminating rules in fundamentally rule-governed domains. We argue that with a suitable choice of high-level, rule governed task, representation, processing architecture, and learning algorithm, neural networks

[1] We say that data are noisy when some of its variables are corrupted or missing altogether.

> can represent and learn rules involving higher-level categories while simultaneously learning those categories. The resulting network can exhibit better learning and task performance than neural networks that do not incorporate rules, and have capabilities that go beyond that of purely symbolic rule-learning algorithms." Paul Smolensky [MMS92]

According to Minsky, "Both kinds of intelligent computational systems, symbolic and connectionist, have virtues and deficiencies. It is very important to integrate them, through neural-symbolic systems, in order to explore the capabilities each one possesses"[Min91]. In this sense, the aim of such integration is twofold: while logic can benefit from neural networks' successful applications on various knowledge domains, neural network learning and generalisation processes can be rigorously studied and explained by logic.

> "There is still a feeling that something is wrong with agent systems and artificial intelligence systems even in cases where the system gives the right answer. There are cases where the 'human computer', slow as it is, gives the correct answer immediately while the agent system may take some time to find it. Something must be wrong. Why are we faster? Is it the way we perceive the rules as opposed to the way we represent them in the agent system? Do we know immediately which rule to use? We must look for the 'correct' representation in the sense that it mirrors the way we perceive and apply the rules. Humans use an overall impression to make decisions about how to go about finding answers to queries. Neural networks are in a better position to model this capacity. In fact, we believe that every agent system should have a neural net component." Dov Gabbay [Gab98]

It would certainly be rewarding if the gap between the study of (symbolic) artificial intelligence and the study of (numerical) artificial neural networks could be reduced. This might suggest massively parallel formal systems that prescribe how to reason, in a certain domain, in a way that is enlightening from the point of view of actual practice. On the other hand, logic may be a very useful tool in helping to explain neural networks' inference process, as well as in formalising their learning and generalisation mechanisms.

1.2 Strategies of Neural-Symbolic Integration

According to [Hil95], *Neural-Symbolic Systems* can be divided into: *Unification Systems* and *Hybrid Systems* (see Figure 1.1). The first category comprises connectionist systems that perform some kind of symbolic computation. The second category contains systems that present a logical as well as a connectionist component, which interact with each other.

The principle underlying *Unification Strategies* is that all the functionality of symbol processing arises from neuronal structures and processes. Unification strategies comprise *Neuronal Modelling* (or Neuroscience) and *Connectionist Logic Systems*. The first group investigates the relationship between specific cognitive tasks and biological reality. By developing computational models of the cognitive tasks of the brain, it attempts to understand how the brain works by building on its cellular units: the neurons [OM00]. The second group, Connectionist Logic Systems, is concerned about the development of models of artificial neural networks that can compute complex symbolic processes in parallel. In this group, the representation of symbolic knowledge in neural networks can be either *localised* or *distributed*. In a localised representation, each neuron is a concept, while in a distributed representation, the most elementary concepts arise from the interaction of many processing elements. In both cases, Connectionist Logic Systems might use either *energy minimisation* or *propagation of activation* as the mechanism of inference. Examples of Connectionist Logic Systems by energy minimisation are [Bal86, Pin95, NR92, Vin94]. Among the Connectionist Logic Systems by propagation of activation are [Sha88, HK92, Sun95, HK94].

Differently from Connectionist Logic Systems, *Hybrid Systems*, in general, combine a symbolic component with a connectionist one. Hybrid Systems can be classified in many different ways: by the application domain; the symbolic and connectionist models used; the functionality of the symbolic and neural components of the system; etc. Following Medsker [Med94], we use the degree of interaction between the symbolic and neural components of the system as a classification scheme for Hybrid Systems.

1. *Stand-Alone Models*: There is no interaction between the symbolic and neural components. Both can be used in the same application in order to compare efficiency.
2. *Loosely Coupled Models*: There is a weak interaction between the components. Neural and Symbolic modules perform specific tasks within the system and communicate via data files. We include here, for example, the systems in which a neural network pre-processes data to be used by an expert system.
3. *Tightly Coupled Models*: There is a strong interaction between the components. Communications between the neural and symbolic modules of the system occur via data structures stored in memory.
4. *Fully Integrated Models*: Data structures and processing are not shared between the components by function, but are part of a unique system with a dual (neural and symbolic) nature. Gallant's Connectionist Expert System [Gal88] is the seminal work towards fully integrated models. Other examples include [GO93, Fu91, MR91, KP92, TS94a].

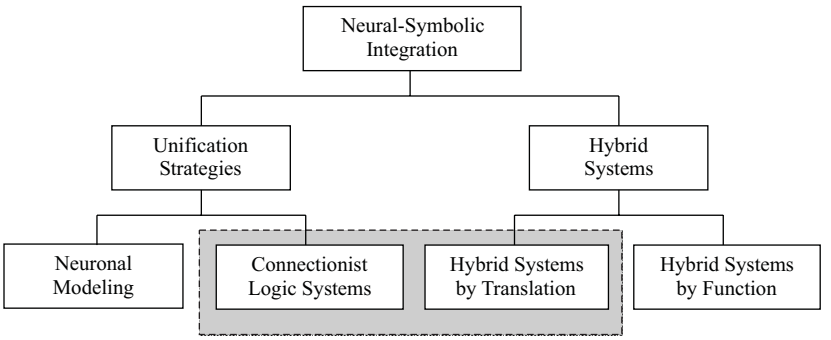

Fig. 1.1. A classification of neural-symbolic systems.

Stand-Alone, Loosely Coupled and Tightly Coupled models all belong to the group of *Hybrid Systems by Function*, according to Hilario's classification [Hil95] (see Figure 1.1). Fully Integrated models, on the other hand, fall into the class of *Hybrid Systems by Translation*. Similarly to Connectionist Logic Systems, Hybrid Systems by Translation use, in general, translation methods from symbolic knowledge (usually production rules) to connectionist models and vice versa. However, while Connectionist Logic Systems are mainly concerned about performing (high-level) symbolic computation in massively parallel models, Hybrid Systems by Translation are concerned about the use of symbolic knowledge to help the process of (low-level) inductive learning, by serving either as background knowledge or as an explanation for the learning method applied. In other words, Connectionist Logic Systems are mainly concerned about the benefits that Neural Networks can bring to Logic, such as massive parallelism, while Hybrid Systems by Translation are rather concerned about the benefits that Logic can bring to Neural Networks, such as learning with background knowledge or rule extraction.

1.3 Neural-Symbolic Learning Systems

In this book, we want to go one step further in the process of neural-symbolic integration, by exploring some of the features of Connectionist Logic Systems (CLSs) and Hybrid Systems by Translation (HSTs). While CLSs are, in general, provably equivalent to a logical formalism, HSTs lack such a fundamental property. On the other hand, CLSs present none or very limited learning capabilities, one of the most important features of HSTs and, indeed, of artificial neural networks. We argue that one can combine features of CLSs and HSTs in order to achieve both equivalence and learning capability in a fully integrated framework. We argue, thus, for the creation of a new category

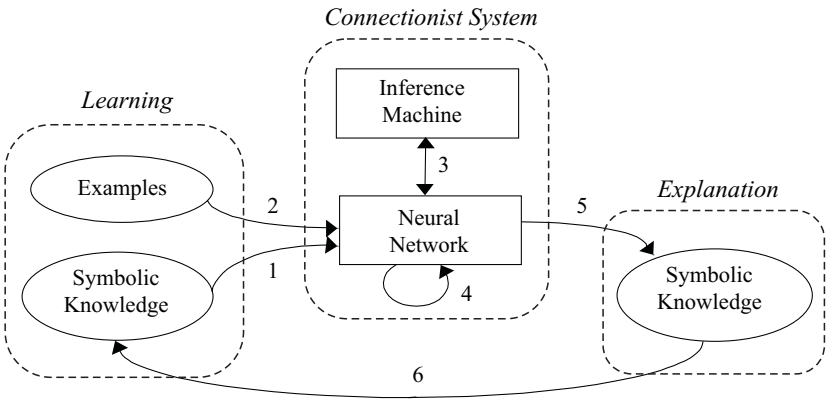

Fig. 1.2. Neural-symbolic learning systems.

in Hilario's classification scheme of Figure 1.1, and call it *Neural-Symbolic Learning Systems*.

Neural-Symbolic Learning Systems contain six main phases: (1) *symbolic knowledge insertion*; (2) *inductive learning with examples*; (3) *massively parallel deduction*; (4) *theory fine-tuning*; (5) *symbolic knowledge extraction*; and (6) *feedback* (see Figure 1.2). In phase (1), a (symbolic) background knowledge is translated into the initial architecture of a neural network by some *Translation Algorithm*. In phase (2), such a network should be able to be trained with examples efficiently, thus refining the initial (incomplete) theory given as *background knowledge*. For example, differently from most CLSs, the network could be trained using the Backpropagation learning algorithm. In phase (3), the network must be able to be used as a massively parallel computational model of the logical consequences of the theory encoded in it. This is so because, as opposed to most HSTs, the Translation Algorithm of a Neural-Symbolic Learning System must be provably correct. In phase (4), the information obtained with the computation carried out in phase 3 may help in fine-tuning the network to represent the knowledge domain better. This mechanism can be used, for example, to solve inconsistencies between the background knowledge and the training examples. In phase (5), the results of refining the network should be explained by the extraction of a revised (symbolic) knowledge from it. As with the insertion of rules, the *Extraction Algorithm* of a Neural-Symbolic Learning System must be provably correct, so that each rule extracted is guaranteed to be encoded in the network. Finally, in phase (6), the knowledge extracted may be analysed by an expert to decide whether it should feed the system once more, closing the learning cycle.

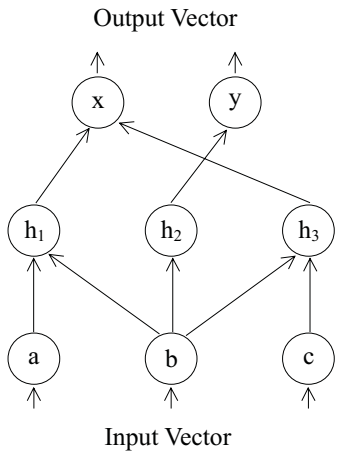

Fig. 1.3. Translating a symbolic knowledge into a neural network.

A typical application of Neural-Symbolic Learning Systems is in safety-critical domains, such as fault diagnosis systems, where the neural network can detect a fault quickly, triggering safety procedures, while the knowledge extracted from it can justify the fault later on. If mistaken, the information can be used to fine tune the learning system.

1.4 A Simple Example

In order to give the reader an overview of the sequence of processes 1 to 6 of Neural-Symbolic Learning Systems (see Figure 1.2), we give a simple illustrative example.

Phase 1: Let us assume that the following incomplete theory is given as background knowledge, in the form of a logic program, $\mathcal{P} = \{a \wedge b \rightarrow x; b \wedge c \rightarrow x; b \rightarrow y\}$. A way of translating $\mathcal{P}$ into a neural network $\mathcal{N}$ is to associate a, b and c with input neurons, and x and y with output neurons. A layer of hidden neurons is then necessary to allow $\mathcal{N}$ to capture the relationships between $\{a, b, c\}$ and $\{x, y\}$ of $\mathcal{P}$. In fact, a hidden neuron of $\mathcal{N}$ for each rule of $\mathcal{P}$ is sufficient. The network should look like that of Figure 1.3, where hidden neuron h_1 should represent rule $a \wedge b \rightarrow x$, hidden neuron h_2 should represent $b \rightarrow y$, and hidden neuron h_3 should represent $b \wedge c \rightarrow x$.

Output neuron y should only be activated if input neuron b is activated. Similarly, output neuron x should only be activated if a and b are activated, or if b and c are activated. In other words, the weights of the connections of the network must be set up such that hidden neuron h_1 performs a logical

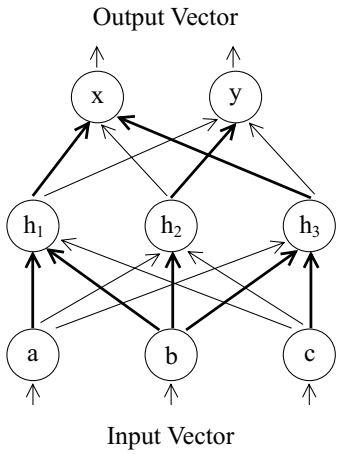

Fig. 1.4. Performing inductive learning with examples.

and between inputs a and b, hidden neuron h_2 is activated if, and only if, b is activated, and hidden neuron h_3 performs a logical *and* between inputs b and c. Similarly, output neuron x must perform a logical *or* between h_1 and h_3, while y should be activated if, and only if, h_2 is activated. If this is the case, we should be able to show that $\mathcal{P}$ and $\mathcal{N}$ are, in fact, equivalent.

Phase 2: Our next step is to perform inductive learning with examples in $\mathcal{N}$. The idea is to change the values of the weights of the network according to a set of training examples (input and output vectors), using some neural learning algorithm. In order to do so, we fully connect $\mathcal{N}$ as in Figure 1.4, so that it can learn new relations between $\{a, b, c\}$ and $\{x, y\}$ in addition to the ones already inserted in it by $\mathcal{P}$. In principle, this process also allows the network to change its background knowledge.

Phase 3: Let us assume that the set of training examples is such that it does not change the background knowledge, but only expands it. Although we do not know yet which is the new knowledge encoded in the network, we should be able to compute its logical consequences in parallel, using the network. For example, the network might have learned the rule $a \wedge c \rightarrow x$. As a result, x could be derived from a and c, as well as from a and b or from b and c. Also, y would still be derivable from b.

Phase 4: Now, suppose that having x and y together is not desirable; that is, suppose that $x \wedge y \rightarrow \bot$ is an integrity constraint of the application domain. The choice between x and y may depend, though, on extra-logical considerations. Assume that, as a matter of fact, x is preferred to y. Neglecting, for the time being, many important aspects of theory revision, the conflict may be adjudicated by evolving the neural network, as depicted in Figure 1.5. The idea is to add a hidden neuron (h_4), responsible for blocking

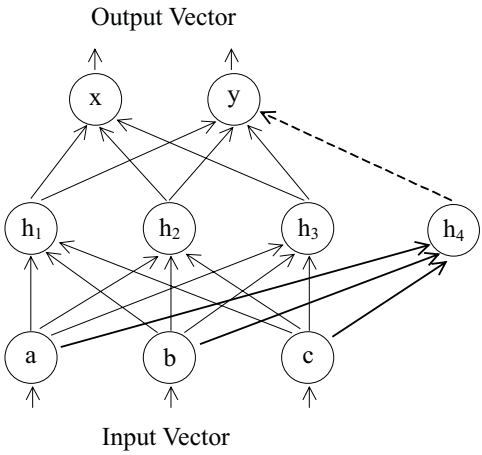

Fig. 1.5. Fine-tuning the network.

the activation of y whenever x is activated. Clearly, the same process could be used to solve inconsistencies of the form $z \wedge \neg z \rightarrow \bot$.

Phase 5: Our next task is to try to extract the (refined) knowledge encoded in the network so that we can explain its answers by inspecting the symbolic knowledge that it encodes. In the case of binary inputs, an option is to use brute force, and check all the possible combinations of input vectors. However, in real-world applications, hundreds of input neurons may be necessary, and the problem may be intractable. The challenge here, therefore, is to consider a small number of input vectors and still make sure that the extraction of rules is correct. In order to do so, we need, in general, to decompose the network into sub-networks, and to extract rules that map $\{a, b, c\}$ into $\{h_1, h_2, h_3, h_4\}$, and $\{h_1, h_2, h_3, h_4\}$ into $\{x, y\}$. However, since a network's behaviour is not equivalent to the behaviour of its parts grouped together, and since neurons h_1, h_2, h_3 and h_4 do not represent concepts, but rules, we need to be especially careful when deriving the final set of rules of a trained network in order to maintain correctness.

We also want to be able to extract rules that reflect the process of generalisation of the network, which occurs during learning, as opposed to rules that account for the network's training set and background knowledge only. For example, a possible generalisation is the rule $a \wedge c \rightarrow x$, which, together with rules $a \wedge b \rightarrow x$ and $b \wedge c \rightarrow x$ of the background knowledge, could be simplified to derive $2(abc) \rightarrow x$, indicating that any two of the concepts a, b and c would imply x. This so-called *M of N* rule could be encoded in neuron h_2, as exemplified in Figure 1.6.

Phase 6: When background knowledge is translated into a neural network, it is possible to create a neat network structure. However, when the

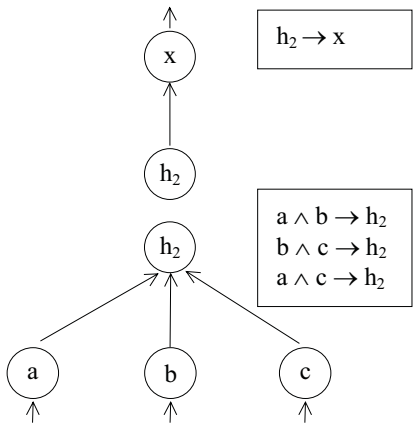

Fig. 1.6. Extracting rules from trained networks.

network is trained with examples, it will most probably lose its well-behaved structure. The task of rule extraction is supposed to discover the knowledge refined by the network. As a result, a new neat network structure could be created by the translation of the new knowledge into the network. In addition, new examples could be trained in such a network, and so on, thus closing the cycle of theory refinement.

In this example, for the sake of simplicity, we have neglected many important aspects of neural-symbolic integration. They will be discussed in detail in the rest of the book. The purpose of this example was to provide the reader with an intuitive presentation of Neural-Symbolic Learning Systems, as well as with an introductory description of the technical material that will follow.

1.5 How to Read this Book

Following a brief presentation of the relevant definitions and results on Inductive Learning, Artificial Neural Networks, Logic Programming and Nonmonotonic Reasoning, and Belief Revision, this book is divided into three parts:

– *Knowledge Refinement in Neural Networks;*
– *Knowledge Extraction from Neural Networks; and*
– *Knowledge Revision in Neural Networks.*

The first part is based on Holldobler and Kalinke's work on CLSs [HK94, HKS99] and on Towell and Shavlik's work on HSTs [Sha96, TS94a, TS94b]. We have carefully chosen the above approaches because Holldobler and

Kalinke, differently from most CLSs, use a neat and simple model of neural networks to compute one of the standard semantics of Logic Programming, thus facilitating the inclusion of learning capabilities into the model, while Towell and Shavlik's Knowledge-based Artificial Neural Networks (KBANN), and its subsequent developments (e.g. [OS93, Opi95]), have been empirically shown to be superior to some of the main neural, symbolic and hybrid systems, being, to the best of our knowledge, the most effective HST to date.

In *Part I*, we will present important theoretical results on Neural-Symbolic Learning Systems – such as the proof of soundness of the Translation Algorithm – and empirically investigate their efficiency by applying them to real world problems of Computational Biology and Fault Diagnosis. We will also compare results with some of the main neural and symbolic systems of Machine Learning.

The second part of the book deals with the problem of symbolic knowledge extraction from trained neural networks. The subject has grown beyond the study of neural-symbolic integration systems and is now a research area on its own (see, for example, [Cra96, Mai98, Mai97, Set97a, Thr94]). Although knowledge extraction is an integral part of Neural-Symbolic Learning Systems, we have tried as much as possible to present the subject independently from the previous chapters. Our approach builds upon Fu's extraction method [Fu94] and some features of KBANN's M of N method [TS93].

In *Part II*, we will present new theoretical results on knowledge extraction from trained neural networks, culminating with the proof of soundness of the Extraction Algorithm, which we believe should be the minimum requirement of any method of rule extraction. We will also present empirical evidence of the performance of the extraction method, by applying it to the problems of Computational Biology and Fault Diagnosis used in Part I to investigate the performance of the learning systems.

The third part of this book tackles the problem of theory revision in neural networks. Theory revision may be necessary as a result of the presence of inconsistencies between a symbolic theory and the result of learning from examples. To the best of our knowledge, this book contains the first account and treatment of the subject.

In *Part III*, we will present a one-shot learning algorithm, which will allow neural networks to evolve incrementally from an undesirable stable state into a new stable state. This method, here called Minimal Learning, can be used to fine-tune the network's answers after learning with examples takes place, but, most importantly, it can be applied to solve inconsistencies in the answers computed by the network. The study of how to deal with inconsistencies can be carried out independently of *Part II* of this book.

We conclude the book by presenting a number of challenges and open problems of neural-symbolic learning systems.

1.6 Summary

We have mentioned that the aim of Neural-Symbolic Integration is twofold: while logic may benefit from the successful application of neural networks on various knowledge domains, the learning and generalisation processes of neural networks may be rigorously studied and explained by logic. We should be able to explore both directions of this "equivalence" relation, taking into consideration aspects such as learning capability and massively parallel deduction and correctness. Notwithstanding this, finding the balance between what we would like to be able to represent in a neural network and what a neural network naturally represents and learns is a difficult task. It all depends on how we want to benefit from the integration of neurons and symbols.

In this monograph, we are committed to high learning performance, for we believe this is the most important asset of artificial neural networks. Thus we concentrate on single hidden layer networks and Backpropagation – the neural learning combination most commonly successfully applied in industry. We then follow the idea of finding the best symbolic representation for such a neural model. The closest match was the class of grounded extended logic programs of Gelfond and Lifschitz [GL91], augmented with the metalevel superiority relations of Nute's Defeasible Logic [Nut94]. The use of more expressive logics, such as first-order logic, would require a proportionally more complex neural model, which could result in a degradation of learning performance. In other words, the limits of synergetic neural-symbolic integration depend on the objectives of the application. Use it to simplify, not to complicate. Be guided by the application and its needs.