International Journal of Neural Systems, Vol. 11, No. 2 (2001) 179–186

NEURAL NETWORKS AND LOGICAL REASONING SYSTEMS: A TRANSLATION TABLE

JOÃO MARTINS
*Laboratório de Mecatrónica, DEEC, IST, Av. Rovisco Pais, 1096 Lisboa Codex, Portugal
Escola Superior de Tecnologia, IPS, Rua do Vale de Chaves, Estefenilha, 2910 Setúbal, Portugal

R. VILELA MENDES*
Grupo de Física-Matemática, Complexo Interdisciplinar,
Univ. de Lisboa, Av. Gama Pinto 2, 1699 Lisboa Codex, Portugal

Received 8 March 1999
Revised 23 January 2001
Accepted 23 January 2001

A correspondence is established between the basic elements of logic reasoning systems (knowledge bases, rules, inference and queries) and the structure and dynamical evolution laws of neural networks. The correspondence is pictured as a translation dictionary which might allow to go back and forth between symbolic and network formulations, a desirable step in learning-oriented systems and multicomputer networks. In the framework of Horn clause logics, it is found that atomic propositions with n arguments correspond to nodes with nth order synapses, rules to synaptic intensity constraints, forward chaining to synaptic dynamics and queries either to simple node activation or to a query tensor dynamics.

1. Introduction

Neural networks and logical reasoning systems, because they both rely on Turing models of computation, are equivalent in the sense that whatever is computable in one framework must also be computable in the other. How to establish the equivalence in each particular case is however a non-trivial and interesting issue, because problems that are addressed in a simple manner in one approach turn out to be intractable in the other and vice-versa. For example, combinatorial optimization problems are efficiently handled by neural networks, whereas logical systems succumb to combinatorial explosion. Conversely, for highly structured problems where appropriate heuristics are known, it is simpler and faster to use a logical reasoning system, the corresponding neural network system taking a long time to tune up, because rule-based information is in general not so easy to build in the network architecture.

In an attempt to relate these two computational models, some authors have tried to identify the nature of the rule-based information that may be extracted from the networks as well as the network structures that correspond to particular logical operations.[1–7] On the other hand the use of hybrid systems has been proposed to solve complex problems.[8–14]

Extraction of rules from networks is an issue of practical importance in the construction of expert systems from example-trained networks. On the other hand when some prior rule-based information is known about a problem but nevertheless a network implementation seems appropriate, it would be useful to have some simple rules to implement the symbolic information on the architecture of the network. Neural networks take advantage of parallel VLSI hardware implementations, which largely improve the processing speed and it is not so clear how to take advantage of similar implementations

for symbolic processing. Therefore even for problems that are naturally stated in logic terms, it might be useful to have a translation dictionary for hardware implementation purposes. The establishment of a concise way to go back and forth between symbolic and network formulations is also welcome in learning-oriented systems and in the design of multicomputer networks. Different architectures make different types of learning easier or harder to design and in multicomputer networks it is essential for algorithms and architectures to fit together as well as possible. Finally and independently of the practical issues, the establishing of a compact translation dictionary between the two paradigms might be a useful step in the development of a unified language for cognitive processes.

Doyne Farmer, in a classical paper,[15] has shown that there is a common framework in which neural networks, classifier systems, immune networks and autocatalytic reaction networks, may be treated in a unified way. The general model, to which all these models are mapped, provides then an extension of the scope of neural network models.[16] In a similar way, providing a bridge between neural networks and logical systems ("a new entry to the Rosetta stone") might suggest new algorithms and applications in both domains.

Past attempts at establishing a logic-networks dictionary concern either the question of the architecture required to represent some types of logical operations or the refining of the numerical part of the knowledge base. In some cases, an extension of the usual connectionist framework is required and a full correspondence is not established. Here we describe an attempt to establish a correspondence involving all the basic elements that are present in logical systems (*knowledge base*, *rules*, *inference*, *recursion* and *handling of queries*).

In a neural network one has a (distributed) memory on the *connection strengths* (*synapses*), a *learning dynamics* on the synapses and an *activation dynamics* of the node intensities. In a logical reasoning system there is some set of *ground facts* about the *objects* in the domain, a set of *rules* which are potential knowledge concerning relations between the objects and an *inference* mechanism (*backward* or *forward chaining*) allowing for the extraction of further information and the answering of *queries*.

It has been argued that trying to isolate, in a network, the structures that correspond to specific logical statements or operations is a waste of time because everything in a network (memory, rules and inference) is distributed everywhere and forever inseparable. This may well be true for some architectures and some classes of concepts. However even the identification of the modular structures that correspond to the logic elements will be a useful step. For example, we conclude here that a natural network representation of an atomic proposition is a node with nth-order synapses. Because of the universal approximation properties of neural networks, that same proposition might as well be represented by first-order synapses, the proposition corresponding then to a submodule of several neurons. However the identification of the kind of minimal submodule that corresponds to a specific logical element is already a useful step. On the other hand, establishment of rules, inference mechanisms and queries are related to dynamical laws in the corresponding network, thus providing a dynamical systems view of the logical reasoning process.

2. The Network Representation of a Logic System

For definiteness, the scope of the logical system that is considered, is a subset of the Prolog logic programming language. Namely the logic system is specified by a set of *Horn clauses* which are constructed from *terms* which are either *constants* or *variables* or *structures*. A constant represents some concrete object in the domain of the problem. It is represented in the logical system by an indecomposable elementary symbol (an *atom*). Structures are restricted to be atomic propositions of the general form

$$\textbf{functor}(\textbf{parameter list}) \tag{1}$$

The functor is an atom and the parameter list is any list of atoms, variables or other atomic propositions. Finally a variable is an entity that can at any time be bound to any atom (constant or functor). Small letters will be used for the atoms and capitals for the variables.

The first step is to find a network representation of the basic entities of the logical system. Each atom will correspond to a *node*. Therefore not only

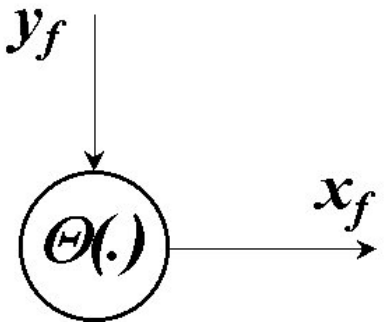

Fig. 1. Network representation of a constant.

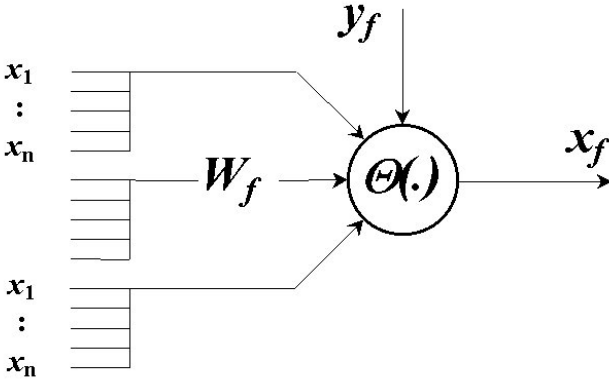

Fig. 2. Network representation of a functor.

constants but also functors will be assigned a node in the network.

The network nodes have output zero or one and threshold one. The nodes are connected into a network, the *inputs* $x_1 \cdots x_n$ being equipped with connection strengths (synapses) $W_{f,i_1,\ldots,i_n}$ of order n[17–19] if n is the number of parameters in the parameter list of the functor. In particular a simple constant has no input. The *node activation* is

$$x_f = \theta\left(\sum_{i_1 \cdots i_n} W_{f,i_1,\ldots,i_n} x_1 \cdots x_n + y_f\right) \qquad (2)$$

where y_f (the *polarization*) is zero unless there is a query involving this node. θ is the Heaviside function. Figure 1 displays the neural network representation of a *constant node* and Fig. 2 that of a *functor node.* The notation $\theta(\cdot)$ is used to denote the Heaviside function with arbitrary arguments.

Whenever in the knowledge base, the relation

$$f(i_1, \ldots, i_n)\,. \qquad (3)$$

is stated as a true fact, the corresponding synaptic strength is

$$W_{f,i_1,\ldots,i_n} = 1 \qquad (4)$$

Otherwise, if the relation f does not hold for these particular set of parameters, the connection strength vanishes. This is the straightforward way to represent in the network the *ground facts* of the knowledge base, that is, the relations that are explicitly stated about the atoms. On the other hand relations and clauses that involve variables correspond to constraints on the remaining connection strengths. This is better explained through an example. Let a knowledge base be

$$\begin{aligned} &f(b,c)\,. \\ &m(a,c)\,. \\ &m(d,a)\,. \\ &p(X,Y) \Leftarrow m(X,Y)\,. \\ &p(X,Y) \Leftarrow f(X,Y)\,. \\ &g(X,Z) \Leftarrow p(X,Y) \wedge p(Y,Z)\,. \end{aligned} \qquad (5)$$

The ground facts (the first three headless clauses) imply

$$W_{fbc} = 1; \quad W_{mac} = 1; \quad W_{mda} = 1 \qquad (6)$$

all other W_{fXY} and W_{mXY} being set to zero. Let the connection strengths take only the values zero and one. Then the last three clauses in (5) translate into the following constraints on the remaining W_{pXY} and W_{gXY} connection strengths

$$\begin{aligned} &W_{pXY} \geq W_{mXY} \\ &W_{pXY} \geq W_{fXY} \\ &W_{gXZ} \geq W_{pXY} W_{pYZ} \end{aligned} \qquad (7)$$

These constraints are established in the network by a parallel dynamical law. Define the potential function

$$\begin{aligned} V_1 = &\frac{1}{2}\sum_{XY} \chi(W_{mXY} - W_{pXY}) \\ &+ \frac{1}{2}\sum_{XY} \chi(W_{fXY} - W_{pXY}) \\ &+ \frac{1}{2}\sum_{XYZ} \chi(W_{pXY} W_{pYZ} - W_{gXZ}) \end{aligned} \qquad (8)$$

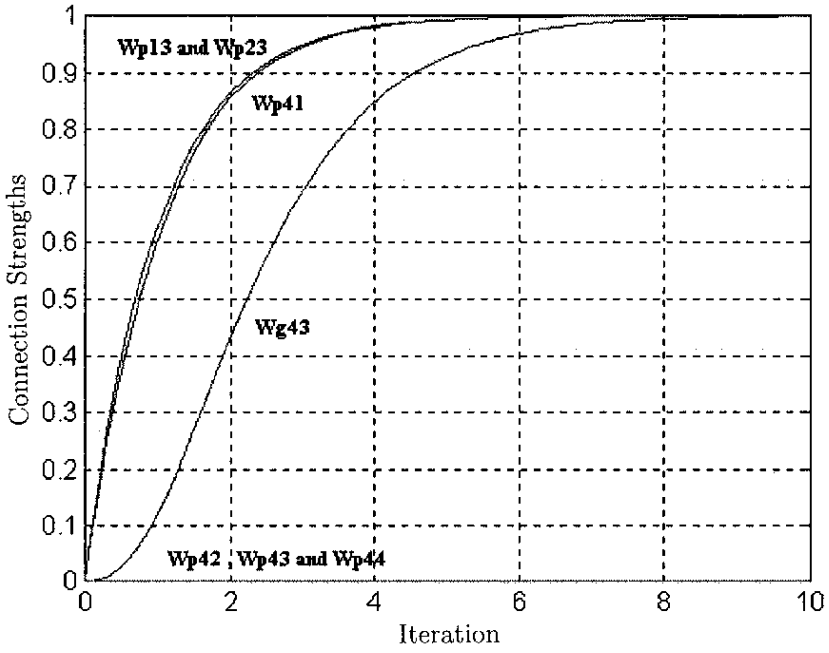

Fig. 3. A connection strengths evolution example.

where $\chi(x) = x^2\theta(x)$. The strengths (6), fixed by the ground statements, are considered as frozen and the others evolve in parallel according to gradient descent dynamics

$$\frac{dW_{pXY}}{dt} = -\frac{\partial V_1}{\partial W_{pXY}}; \quad \frac{dW_{gXY}}{dt} = -\frac{\partial V_1}{\partial W_{gXY}} \quad (9)$$

This dynamical law drives the connection strengths to values zero or one satisfying the constraints (7). Figure 3 shows an example of dynamical evolution of the connection strengths.

The method used in this example holds in general and for each rule of the form

$$g(X_1, \ldots, X_n) \Leftarrow \prod_i \wedge f_i(Y_1, \ldots, Y_{p_i}) \quad (10)$$

a term

$$\frac{1}{2} \sum_{\{X\}\{Y\}} \chi \left(\prod_i W_{f_i Y_1 \cdots Y_{p_i}} - W_{gX_1 \cdots X_n} \right) \quad (11)$$

is added to the potential V_1, the sum being over all the variable terms.

So far we have established a correspondence between atomic propositions with n parameters and nth-order nodes and between rules and a parallel dynamics of the synapses. Because the synapse dynamics is equivalent to a forward chaining step, the network now contains on its connection strengths all the available information about the problem. Therefore information may be retrieved by simply looking at the values of the connection strengths. Alternatively an explicit scheme to answer queries may be established in the following way.

We may consider two different types of queries, simple queries without variables and queries with variables. In the first case, for example

$$? \Leftarrow g(a, b) \quad (12)$$

the question is to check whether the relation g holds for the atoms a and b. In this case it suffices to excite the polarizations y_a and y_b of the nodes a and b and see whether the node g lights up. The same happens for more complex queries like, for example

$$? \Leftarrow g(h(a, b), c) \wedge f(c, d) \quad (13)$$

where now the polarizations to be excited are y_a, y_b, y_c and y_d and the answer to the query is positive if nodes g and f light up simultaneously.

For queries involving variables, for example

$$? \Leftarrow m(d, X) \wedge p(X, c) \quad (14)$$

the question is to find the instantiations of X, if any, that make this a true statement. The product of the connection strengths $W_{mdX}W_{pXc} = \alpha_X$ defines a vector α_X in atom space and the non-zero entries of the vector are the instantiations that satisfy the query. A similar scheme holds for queries involving several variables. For example, for

$$? \Leftarrow g(h(X,c),a) \wedge f(c,Y) \tag{15}$$

ones defines a tensor $\beta_{XY} = W_{gha}W_{hXc}W_{fcY}$, the non-zeros entries of which are the positive answers to the query. With all the connection strengths established in the network by the (forward chaining-like) dynamics of Eq. (9) it might be an easy matter (depending on the type of implementation used) to look up the non-zero entries of α_X or β_{XY}. Alternatively one may regard the network connection strengths as potential knowledge and generate dynamically the non-zero entries of the tensors

$$\frac{d\alpha_X}{dt} = -\frac{\partial V_2}{\partial \alpha_X}; \quad \frac{d\beta_{XY}}{dt} = -\frac{\partial V_3}{\partial \beta_{XY}} \tag{16}$$

with

$$\begin{aligned} V_2 &= \sum_X (W_{mdX}W_{pXc} - \alpha_X)^2 \\ V_3 &= \sum_{XY} (W_{gha}W_{hXc}W_{fcY} - \beta_{XY})^2 \end{aligned} \tag{17}$$

Regarding the network structure and connection strengths as potential knowledge (as opposite to explicit knowledge), the dynamics of the query tensors defined above plays a role similar to backward chaining in logical programming. As in (10)–(11) the generalization of this dynamics, in the space of query tensors, to general queries is straightforward.

The whole construction is summarized in Table 1.

As a simple example we discuss problem 148 of Ref. 24, originally suggested in "The Thinking Computer" by Bertrand Raphael. It considers the following functors:

$number(X,N)$	means that person X can be reached by calling phone number N.
$visits(X,Y)$	means that person X is visiting person Y.
$at(X,Y)$	means that person X is at the residence of person Y.
$phone(X,N)$	means that person X has phone number N.

and the following clauses:

$$\begin{aligned} at(X,Z) &\Leftarrow visits(X,Y) \wedge\ at(Y,Z) \\ number(U,N) &\Leftarrow at(U,V) \wedge\ phone(V,N)\,. \end{aligned}$$

The purpose is to reach a person, having a phone number data base and knowing who visits whom and where the visited person is.

Consider the following ground facts:

$$\begin{aligned} phone(coleman, \text{'100001'})\,. \\ phone(gordon, \text{'100002'})\,. \\ phone(wagner, \text{'100003'})\,. \\ phone(smith, \text{'100004'})\,. \\ visits(coleman, wagner)\,. \\ at(wagner, gordon)\,. \end{aligned}$$

The clauses in this example define the following potential

$$\begin{aligned} V_4 = &\frac{1}{2}\sum_{XYZ} \chi(W_{visitsXY}W_{atYZ} - W_{atXZ}) \\ &+ \frac{1}{2}\sum_{UVN} \chi(W_{atUV}W_{phoneVN} - W_{numberUN}) \end{aligned} \tag{18}$$

Table 1. Logical system — Neural network translation table.

Logical system	Network
Constant	Single node
Atomic proposition (n arguments)	Node with nth-order synapses
Rules	Synaptic intensity constrains
Rule application (forward chaining)	Synaptic dynamics
Queries	Node activation/Query tensor dynamics

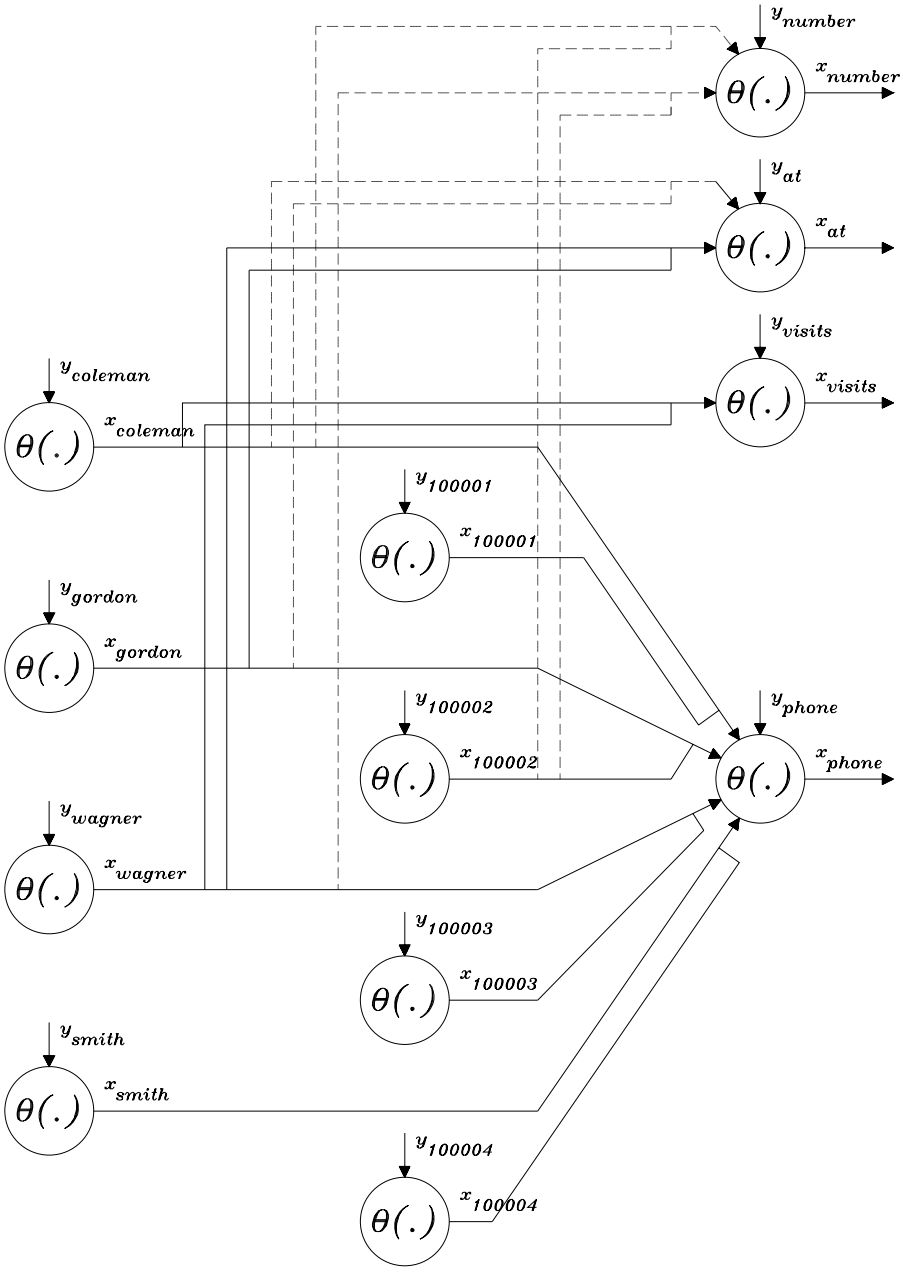

Fig. 4. Neural network representation of problem 148 in Ref. 24.

The evolution of the connection strengths, according to the gradient descendent dynamics, leads to the neural network in Fig. 4, where only the non-zero connections strengths are shown. The continuous lines represent the connection strengths of the ground facts and the dashed ones the connection strengths obtained from the clauses used to construct the potential V_4.

3. Conclusion and Remarks

The construction described before, has emphasized the establishment of a correspondence between the pure logic elements of logic programming and both the hardware and dynamical operations of neural networks. Logic programming systems however, in addition to its pure logic elements, involve some

non-logical features like the handling of arithmetics and procedural prescriptions for computing efficiency. These features need not concern us here because numerical capabilities are native to networks and procedural rules are implementation dependent.

A more serious limitation concerns the restriction to Horn clauses. Horn clause logic is quite expressive and of wide computing power. However it cannot handle negation in a satisfactory way, especially for non-exhaustive databases. In some cases it is acceptable to reduce, by renaming,[20] the set of non-Horn to a Horn clause set. However, depending on the nature of the database, it may be more convenient to retain a certain degree of non-Horness and to establish an inference procedure through case analysis.[21–23] At a formal level, and to some extent, our formalism might be extended to non-Horn clauses. For example, if a non-Horn clause is

$$h(X,Z) \vee g(X,Z) \Leftarrow p(X,Y) \wedge p(Y,Z) \qquad (19)$$

the relation for the connection strengths that extends (7) is

$$W_{hXZ} + W_{gXZ} \geq W_{pXY} W_{pYZ} \qquad (20)$$

However it is not clear, at this time, how to implement dynamically, in an elegant manner, the case analysis mechanism.

In this paper, we have been concerned with logical reasoning systems in its sharp logics version. A natural concern is the extension to reasoning with uncertainty. Given that the handling of real numbers is quite natural in networks, an extension where all quantities (node activations and connection strengths), instead of being 0 or 1, take a continuous range of values, does not seem to raise any special difficulties. Notice however that the straightforward extension of our network operations, as defined in Sec. 2, leads to a representation of the fuzzy intersection by a product rather than by the min operation.

References

1. S. I. Gallant 1988, "Connectionist expert systems," *Commun. of the ACM* **31**, 152–169.
2. S. I. Gallant 1993, *Neural Network Learning and Expert Systems* (MIT Press, Cambridge, Massachusetts).
3. G. Towell and J. Shavlik 1993, "The extraction of refined rules from knowledge-based neural networks," *Machine Learning* **13**, 71–101.
4. L.-M. Fu 1994, "Rule generation from neural networks," *IEEE Trans. on Systems, Man and Cybernetics* **24**, 1114–1124.
5. I. K. Sethi and J. H. Yoo 1996, "Symbolic mapping of neurons in feedforward networks," *Pattern Recognition Lett.* **17**, 1035–1046.
6. M. Botta, A. Giordana and R. Piola 1997, "Refining first order theories with neural networks," in *Foundations of Intelligent Systems*, eds. Z. W. Raś and A. Skowron, Lecture Notes in Artificial Intelligence 1325 (Springer, Berlin), pp. 84–93.
7. R. Sun 1992, "On variable binding in connectionist networks," *Connection Science* **4**, 93–124.
8. R. Sun 1993, *Integrating Rules and Connectionism for Robust Reasoning* (Wiley, New York).
9. V. Honavar and L. Uhr 1994, *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration* (Academic Press, San Diego) and references therein.
10. J. A. Bardnen 1991, "Encoding complex symbolic data structures with some unusual connectionist techniques," in *Advances in Connectionist and Neural Computation Theory, Vol. 1: High Level Connectionist Models*, eds. J. A. Barnden and J. B. Pollack (Ablex, Norwood), pp. 180–240.
11. J. A. Barnden and J. B. Pollack (eds.) 1991, *Advances in Connectionist and Neural Computation Theory, Vol. 1: High Level Connectionist Models* (Ablex, Norwood).
12. L. Shastri 1990, "Connectionism and the computational effectiveness of reasoning," *Theoretical Linguistics* **16**, 65–87.
13. J. B. Pollack 1990, "Recursive distributed representations," *Artificial Intelligence* **46**, 77–105.
14. S. Pinker and A. Prince 1988, "On language and connectionism: Analysis of a parallel distributed processing model of language acquisition," *Cognition* **28**, 73–193.
15. J. D. Farmer 1990, "A Rosetta stone for connectionism," *Physica* **D42**, 153–187.
16. J. A. Dente and R. Vilela Mendes 1996, "Unsupervised learning in general connectionist systems," *Network: Computation in Neural Systems* **7**, 123–139.
17. W. R. Softy and D. M. Kammen 1991, "Correlations in high dimensional or asymmetric data sets: Hebbian neuronal processing," *Neural Networks* **4**, 337–347.
18. J. G. Taylor and S. Coombes 1993, "Learning higher order correlations," *Neural Networks* **6**, 423–427.
19. J. W. Clark, K. A. Gernoth and M. L. Ristig, *High-order probabilistic perceptrons as Bayesian inference engines*, Trieste preprint IC/94/236, 1994.
20. X. Nie and Q. Guo 1997, "Renaming a set of non-Horn clauses," in *Foundations of Intelligent Systems*, eds. Z. W. Raś and A. Skowron, Lecture Notes in Artificial Intelligence 1325 (Springer, Berlin), pp. 600–608.

21. D. W. Loveland 1991, "Near-Horn Prolog and beyond," *Journal of Automated Reasoning* **7**, 1–26.
22. D. A. Plaisted 1982, "A simplified problem reduction format," *Artificial Intelligence* **18**, 227–261.
23. D. A. Plaisted 1988, "Non-Horn clause logic programming without contrapositives," *Journal of Automated Reasoning* **4**, 287–325.
24. H. Coelho and J. C. Cotta 1988, *Prolog by Example* (Springer-Verlag).