A Neural Implementation of Conceptual Hierarchies with Bayesian Reasoning*

Pekka Orponen† Patrik Floréen Petri Myllymäki Henry Tirri

Department of Computer Science, University of Helsinki
SF–00510 Helsinki, Finland

Abstract

We present a scheme for translating high-level descriptions of conceptual hierarchies into a neural network representation. The intuitive semantics of a conceptual hierarchy is provided by a Bayesian net, and the neural network implementation provably approximates the behaviour of this net under a stochastic simulation rule.

1 Introduction

Hybrid neural–symbolic programming systems have recently been attracting increasing attention as a potentially productive symbiosis of two complementary methodologies, benefiting from the advantages and avoiding the disadvantages of both. In a hybrid system, the symbolic component would be used for complicated inference, while the neural component would provide a basic robustness to the knowledge representation. Some recent approaches to developing such systems are reported in [1, 2, 3, 4, 6, 7, 9].

In this paper, we describe a novel knowledge representation scheme which is currently being implemented in the hybrid NEULOG system, developed at the University of Helsinki [6]. In our scheme, a conceptual hierarchy, described in a high-level language, is interpreted as defining a particular probability distribution over a set of variables that correspond to the concepts and their possible attribute values in the network. This probability distribution can be conveniently represented as a Bayesian network [8], and this representation has a natural realization as a neural net. The neural realization will be "correct" in the sense that it provably approximates the behaviour of the Bayesian network under a stochastic simulation rule (cf. [8, Section 4.4]).

Of the hybrid system approaches mentioned above, ours is closest to that of Shastri [9]. In Shastri's system, a conceptual hierarchy is similarly interpreted as defining a particular probability distribution, and the implementation network provably performs evidential reasoning according to this distribution. The fundamental difference to our system is that Shastri does not make the connection to Bayesian networks, and so instead of a stochastic Bayesian computation, his system implements a "maximum entropy" reasoning rule. As a consequence of this difference, Shastri's networks require more powerful computing elements and a more involved control regime than ours.

2 Bayesian Networks and Stochastic Simulation

We present here a brief summary of Bayesian nets; for an extensive discussion, see Pearl's excellent book [8]. Let P be a probability distribution over the variables $X_1, \ldots, X_n$. For simplicity, let us assume that all the variables are binary-valued. As in [8], we use boldface symbols to denote sets of variables, together with the following concise notation: if $\boldsymbol{X} = \{X_{i_1}, \ldots, X_{i_k}\}$ is a set of variables, $\boldsymbol{X} = \boldsymbol{x}$ means that $\boldsymbol{x}$ is a k-bit binary vector and $X_{i_j} = \boldsymbol{x}(j)$ for every $j = 1, \ldots, k$.

*This work has been supported by the Academy of Finland, and by the Finnish Technology Development Center (TEKES) under its FINSOFT programme.

†Current address of first author: Dept. of Computer Science, Univ. of Toronto, Toronto, Ont. M5S 1A4, Canada.

Definition 2.1 [8, p. 83] Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be sets of variables. Then $\boldsymbol{X}$ is *conditionally independent of* $\boldsymbol{Y}$*, given* $\boldsymbol{Z}$, if

$$P(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z} = \boldsymbol{z}) = P(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{Z} = \boldsymbol{z})$$

holds for all vectors $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ such that $P(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z} = \boldsymbol{z}) > 0$.

Intuitively, the variables in $\boldsymbol{Z}$ intercept any causal connections between the variables in $\boldsymbol{X}$ and the variables in $\boldsymbol{Y}$: knowing the values of $\boldsymbol{Z}$ renders information about the values of $\boldsymbol{Y}$ irrelevant to determining the distribution of $\boldsymbol{X}$.

Definition 2.2 [8, p. 120] A *Bayesian network* for a probability distribution P is a directed acyclic graph D whose nodes correspond to the variables $X_1, \ldots, X_n$, and whose topology satisfies the following: each variable X is conditionally independent of all its non-descendants in D, given its set of parents $\boldsymbol{F}_X$, and no proper subset of $\boldsymbol{F}_X$ satisfies this condition.

The parent variables $\boldsymbol{F}_X$ may intuitively be thought of as the immediate causes of variable X. The importance of a Bayesian network structure lies in the way the network facilitates computing conditional probabilities. To make this precise, let us introduce the following notation: given a variable X in a Bayesian network D, let $\boldsymbol{F}_X$ denote the set of parents of X, $\boldsymbol{S}_X$ the set of children of X, and $\boldsymbol{U}_X$ the set of all variables except X. The following result is then an immediate consequence of Definition 2.2:

Theorem 2.1 [8, p. 121] *Given a variable X in a Bayesian network D, define*

$$\boldsymbol{B}_X = \boldsymbol{F}_X \cup \boldsymbol{S}_X \cup \bigcup_{Y \in \boldsymbol{S}_X} \boldsymbol{F}_Y.$$

Then X is conditionally independent of $\boldsymbol{U}_X - \boldsymbol{B}_X$, given $\boldsymbol{B}_X$. $\square$

Thus, to determine the distribution of a variable X, given the values of all the other variables, it suffices to consider only the values of the variables in $\boldsymbol{B}_X$. A set of variables covering X in this sense is called a *Markov blanket* of X [8, p. 97]. Even an explicit formula for computing the distribution of X from its Markov blanket can be given.

Theorem 2.2 [8, p. 218] *Let D be a Bayesian network over a variable set $\boldsymbol{U} = \{Y_1, \ldots, Y_n\}$. Given any value vector $\boldsymbol{u} : \{1, \ldots, n\} \to \{0, 1\}$ and a set of variables $\boldsymbol{Z} \subseteq \boldsymbol{U}$, let $\boldsymbol{z}$ denote the restriction of $\boldsymbol{u}$ to the domain $\{i : Y_i \in \boldsymbol{Z}\}$. The probability distribution of each variable $X = Y_i$ in the network, conditioned on the values of all the other variables, may then be expressed as:*

$$P(X = x | \boldsymbol{U}_X = \boldsymbol{u}_X) = cP(X = x | \boldsymbol{F}_X = \boldsymbol{f}_X) \prod_{Y = Y_j \in \boldsymbol{S}_X} P(Y = \boldsymbol{u}(j) | \boldsymbol{F}_Y = \boldsymbol{f}_Y), \tag{1}$$

where x is 0 or 1, c is a constant independent of x, and $\boldsymbol{u}$ is any value vector for $\boldsymbol{U}$ such that $\boldsymbol{u}(i) = x$. $\square$

Based on this result, and the fundamental properties of Markov chains [5], Pearl proposes a *stochastic simulation* technique for computing probability assignments in a Bayesian network. Assume that we are given as initial data the value vector $\boldsymbol{y}$ for some set of variables $\boldsymbol{Y}$. Then the conditional probabilities $P(X = 1 | \boldsymbol{Y} = \boldsymbol{y})$ can be estimated by the following procedure: first instantiate all variables in $\boldsymbol{U} - \boldsymbol{Y}$ to some arbitrary initial values; then repeatedly choose one variable $X \in \boldsymbol{U} - \boldsymbol{Y}$ and assign to it a new value in accordance with (1), given the current values of the variables in $\boldsymbol{B}_X$. More precisely, when a variable X is being considered in a configuration where the other variables have values $\boldsymbol{U}_X = \boldsymbol{u}_X$, first the ratio

$$r = \frac{P(X = 1 | \boldsymbol{U}_X = \boldsymbol{u}_X)}{P(X = 0 | \boldsymbol{U}_X = \boldsymbol{u}_X)},$$

is computed according to (1). (Note that the expression for r no longer involves the normalization constant c.) Then X is assigned a new value at random, with value 1 having probability $p = r/(1 + r)$, and value 0 having probability $1 - p = 1/(1 + r)$. Provided that all the probabilities $P(Z = z | W = w)$ associated with the arcs in the Bayesian network are nonzero, the theory of Markov chains guarantees that the frequency with which $X = 1$ in the simulation converges to the correct probability value $P(X = 1 \mid \boldsymbol{Y} = \boldsymbol{y})$.

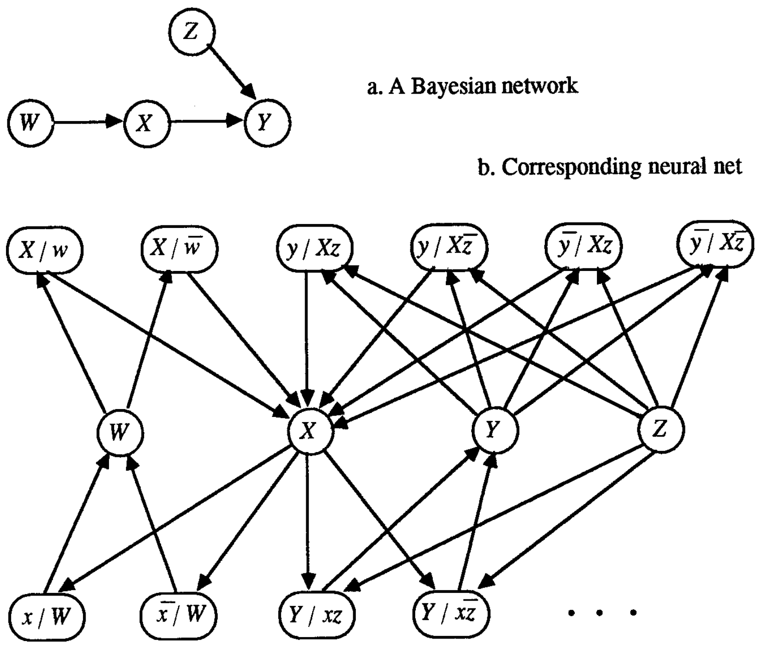


Figure 1: A simple Bayesian network and part of its neural realization.

3 A Neural Realization

In this section we show how Pearl's stochastic simulation scheme for Bayesian nets can be realized by a neural network consisting of two layers of simple binary stochastic units.

The first layer of units represents the variables in the Bayesian network. These are called the *variable units*. The second layer, called the *context units*, contains for each variable X one unit for

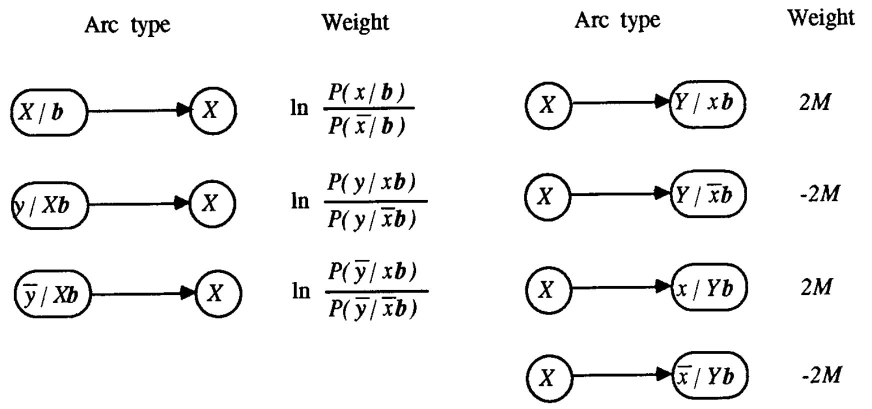
1. each possible vector of values $\boldsymbol{f}_X$ for the variables in $\boldsymbol{F}_X$;
2. each possible vector of values $\boldsymbol{f}_Y^*$ for the variables in $\boldsymbol{F}_Y^* = \boldsymbol{F}_Y \cup \{Y\} - \{X\}$, for each variable $Y \in \boldsymbol{S}_X$.

The stochastic activation function of each unit is the sigmoid $\sigma(t) = (1 + e^{-t})^{-1}$, where t is the net weighted input to the unit.

Figure 1 shows a simple Bayesian network and a part of its neural realization. Note the following shorthand used in the figure: for a variable V, v denotes the value assignment $V = 1$, and $\bar{v}$ denotes the value assignment $V = 0$. We shall return in Section 4 to the very important issue of how the exponential growth in the number of context units can be avoided in many practical situations.

The variable units simulate the behaviour of the variable nodes in a Bayesian network under the stochastic simulation scheme. The context units associated with a given variable X collect information about the current states of those variable units whose variables belong to the Markov blanket $\boldsymbol{B}_X$ of X. The variable and context unit layers are updated alternately, so that in one phase all the variable units in parallel assume new states, and in the following phase all the context units in parallel assume the appropriate states to propagate the new information concerning the variable units.

The rules for assigning arc weights in the neural network are summarized in Figure 2. The constant M appearing in the rules should be chosen so large that $\sigma(M)$ is "sufficiently close" to 1 and $\sigma(-M)$ is sufficiently close to 0. In addition to the rules shown in Figure 2, each context unit of the form $(X|\boldsymbol{b})$ (resp. $(b_1|X\boldsymbol{b}'))$ receives a constant input of magnitude $-(2p-1)M$, where p is the number of variables occurring positively in the vector $\boldsymbol{b}$ (resp. $b_1\boldsymbol{b}'$).



Arc type
Weight
Arc type
Weight
X / b
X
$\ln \frac{P(x/b)}{P(\bar{x}/b)}$
y / Xb
$\ln \frac{P(y/xb)}{P(y/\bar{x}b)}$
$\bar{y}$ / Xb
$\ln \frac{P(\bar{y}/xb)}{P(\bar{y}/\bar{x}b)}$
Y / xb
2M
Y / $\bar{x}$b
-2M
x / Yb
2M
$\bar{x}$ / Yb
-2M

Figure 2: Assignment of weights to arcs in the neural realization.

The effect of the input arc weights on the context units is to turn on with high probability (depending on the choice of M) exactly those units whose $\boldsymbol{b}$ vectors are consistent with the current states of the variable units. The arc weights from context units to variable units implement formula (1). To illustrate this, consider for instance the updating of variable unit X in Figure 1 in a configuration where the units belonging to its Markov blanket have states $W = 1, Y = 0, Z = 1$. In the context update phase, with high probability exactly the context units $(X|w)$ and $(\bar{y}|Xz)$, among the set of units providing input to X, are activated. Hence in the variable update phase, unit X receives a net input of magnitude

$$t = \ln \frac{P(x|w)}{P(\bar{x}|w)} + \ln \frac{P(\bar{y}|xz)}{P(\bar{y}|\bar{x}z)}.$$

Denoting

$$r = \frac{P(X = 1|w\bar{y}z)}{P(X = 0|w\bar{y}z)}$$

we see, by formula (1), that $t = \ln r$, and so X assumes the value 1 with the correct probability of

$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{1}{1+r^{-1}} = \frac{r}{1+r}.$$

4 Translating Conceptual Hierarchies into Bayesian and Neural Nets

We now outline a scheme for translating high-level descriptions of conceptual hierarchies into a Bayesian network representation, which may then be realized neurally as described in the previous section. For simplicity, we consider here only conceptual hierarchies that are *trees*, i.e. such that every concept has at most one immediate ancestor in the inheritance ordering. However, our techniques can be generalized to arbitrary *singly connected* hierarchies, where there is only one inheritance path from a concept to any of its ancestors. A particularly important feature of our translation scheme is the way the conceptual hierarchy is used to reduce the number of context units in the neural implementation.

As an example, consider the following conceptual hierarchy, described in a generic high-level language:

```
concept fruit is basic (100) with
    taste  : [ sweet (60), sour (40) ];
    colour : { red, green, yellow };
concept lemon is fruit (10) with
    taste  = sour;
    yellow in colour;
    red, green notin colour;
concept apple is fruit (50) with
    origin : [ domestic (25), imported (25) ];
    taste  = sweet (40), sour (10);
    size   = medium;
concept mcintosh is apple (10) with
    origin = domestic;
    taste  = sweet (6), sour (4);
    red in colour;
    green in colour (7);
    yellow notin colour.
```

Here, a description of a single concept consists of a reference to its immediate ancestor (if any), a list of attributes local to this concept and inherited by its descendants, and a sequence of value assignments to attributes visible at the concept. There are two types of attributes: exclusive (indicated by the square brackets "[ ]" enclosing the list of possible values) and set-valued (indicated by the curly braces "{ }"). For any object, an exclusive attribute must be assigned exactly one value from its list of possible values; a set-valued attribute may possess any number of values (including zero) from its list. The parenthesized numbers indicate the "frequency" of a given value for an attribute, or at the header of a concept declaration, the "frequency" of objects falling into that concept class. These numbers may either be actual objective frequencies, or subjective estimates of the "typicality" of certain contingencies.

A Bayesian network corresponding to the "fruits" hierarchy is shown in Figure 3. In this network, a binary variable is assigned to each concept class and attribute–value pair. To save space in the figure, variable names have been abbreviated in an obvious manner: f for "fruit", *t:sw* for "taste = sweet" etc. If variable X is a parent of variable Y in the network, the value of $P(Y|X)$ is indicated next to the arc representing the dependency. The reader can convince him/herself, directly on the basis of Definition 2.2, that the probability distribution determined by the Bayesian net in Figure 3 indeed provides a natural interpretation for the description of the "fruits" hierarchy.

As can be seen from the figure, groups of mutually exclusive variables, such as the immediate descendants of a concept variable (l and a for f), or the value variables for an exclusive attribute (*t:sw* and *t:so*) give rise to zero-probability dependencies in the Bayesian network. Unfortunately, such dependencies cause a problem if the network is queried using a stochastic simulation technique. The convergence of node activity frequencies to their correct values is guaranteed only if all the probabilities associated with the arcs in the network are nonzero.

To circumvent this difficulty, we introduce a parameter η, which is to have some small positive value, and replace all the probability 0 arcs in the network by arcs of probability η. This approximation introduces a small systematic error to the results obtained from the network; the error may of course be decreased at the cost of increased computation time by diminishing η.

Let us then consider realizing the Bayesian fruits network of Figure 3 neurally, according to the scheme of Section 3. It can be seen that because of the hierarchical structure of the network, most of the context units prescribed by the basic translation scheme are in fact redundant: the value combinations indicated in their condition parts can never occur.

Consider, for instance, the node representing the variable *t:so*. According to the basic scheme, this should have input arcs from 16 context units, corresponding to all possible combinations of values to the variables f, l, a, and *t:sw*. Let us for the moment forget about the 0 probability dependency from *t:sw*, and imagine that the parent variables of *t:so* are f, l, and a. Now, by the hierarchy, variables l and a cannot simultaneously have value 1, which reduces the number of context nodes required from 8 to 6. (In these calculations, we are using the exact form of the hierarchy instead of the η approximation.) Moreover, because l and a have no

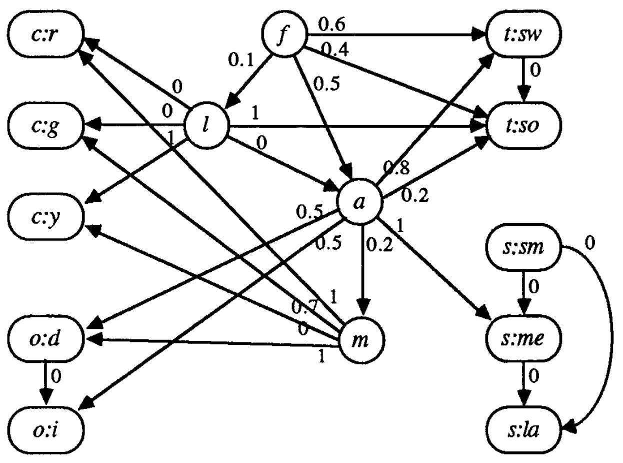


Figure 3: Bayesian network corresponding to the "fruits" description.

other parents than f, knowledge that $l = 1$ or $a = 1$ entails the knowledge that $f = 1$. (Actually, because the variable f is located at the top of the hierarchy, the case $f = 0$ cannot occur anyway.) We obtain immediately the following relations:

$$\begin{aligned} P(t{:}so|f,l,a) & \quad \text{– not possible} \\ P(t{:}so|f,l,\bar{a}) &= P(t{:}so|l) = 1 \\ P(t{:}so|f,\bar{l},a) &= P(t{:}so|a) = 0.2 \end{aligned}$$

The remaining probability value $P(t{:}so|f,\bar{l},\bar{a})$ can be computed most conveniently directly from the textual description of the fruits hierarchy (here $\#c$ denotes the number of objects in concept class c):

$$\begin{aligned} P(t{:}so|f\bar{l}\bar{a}) &= \frac{\#(t{:}so)f\bar{l}\bar{a}}{\#f\bar{l}\bar{a}} = \frac{\#(t{:}so)f - \#(t{:}so)l - \#(t{:}so)a}{\#f - \#l - \#a} \\ &= \frac{40 - 10 - 10}{100 - 10 - 50} = 0.5. \end{aligned}$$

In general, it can be seen that if an attribute-value variable X in the Bayesian network has as parents the concept variables $C_1, \ldots, C_n$, all from the same treelike conceptual hierarchy, then:

1. If C_i and C_j belong to different branches of the tree, then $P(c_i, c_j, \boldsymbol{b}) = 0$ for any value assignment $\boldsymbol{b}$. Hence no context units are needed for value assignments that contain positive occurrences of variables from different branches. Moreover, if C_i and C_j belong to different branches, then $P(x|c_i, \bar{c_j}, \boldsymbol{b}) = P(x|c_i, \boldsymbol{b})$.
2. If C_i and C_j belong to the same branch in the tree, and C_i is an ancestor of C_j, then $P(\bar{c_i}, c_j, \boldsymbol{b}) = 0$ for any value assignment $\boldsymbol{b}$. Hence no context units are needed for value assignments that contain a negative occurrence of some variable, and a positive occurrence of its descendant. Moreover, if C_i is an ancestor of C_j, then $P(x|c_i, c_j, \boldsymbol{b}) = P(x|c_j, \boldsymbol{b})$.

By these two rules, the only context units needed to transmit information from variable units representing concepts to variable units representing attribute values are of the types $(X|c)$ and $(X|c\bar{d}_1 \ldots \bar{d}_k)$, where X is an attribute-value variable, C is a concept variable, and $D_1, \ldots, D_k$ are concept variables descendant to

C. The probabilities $P(x|c)$ can be deduced immediately from the description of the conceptual hierarchy, and the probabilities $P(x|c\bar{d}_1 \ldots \bar{d}_k)$ can be computed as in the example above:

$$P(x|c\bar{d}_1 \ldots \bar{d}_k) = \frac{\#xc\bar{d}_1 \ldots \bar{d}_k}{\#c\bar{d}_1 \ldots \bar{d}_k} = \frac{\#xc - (\#xd_1 + \cdots + \#xd_k)}{\#c - (\#d_1 + \cdots \#d_k)}.$$

So far, we have glossed over a significant complication in the translation scheme: the treatment of mutually exclusive variable sets. Recall that if $Y_1, \ldots, Y_n$ is either the set of children of some concept variable, or the set of attribute-value variables for some attribute of an exclusive type, then there is a probability η dependency arc from each variable Y_i to variable Y_j, where $i < j$. Now it would seem that in such a case an exponential (in n) number of context units would be needed, and that horrendously complex expressions would arise in computing the requisite probabilities $P(y_k|\boldsymbol{by})$, where $\boldsymbol{y}$ is a value assignment to $Y_1, \ldots, Y_{k-1}$, and $\boldsymbol{b}$ is a value assignment to some other variables. Luckily, these probabilities can be well approximated as follows (we omit the precise calculations):

1. if $\boldsymbol{y}$ contains a positive occurrence of some Y_i, $i < k$, then $P(y_k|\boldsymbol{by}) = \eta + \mathcal{O}(\eta^2)$;
2. otherwise
$$P(y_k|\boldsymbol{by}) = (1 - \eta\frac{1 - p_k}{p_k})(1 - \eta\frac{1 - (p_{k-1} + p_k)}{p_k}) + \mathcal{O}(\eta^2),$$
where $p_i = P(y_i|\boldsymbol{b}), i = k-1, k$.

Note that again the number of context units needed is greatly reduced.

Acknowledgment

We wish to thank Geoffrey Hinton for proposing, when confronted with our early ideas of translating high level descriptions into neural nets, that we look into Pearl's Bayesian networks for a precise semantics.

References

[1] M. Derthick, Mundane Reasoning by Parallel Constraint Satisfaction. Ph. D. Thesis, TR CMU-CS-88-182, Carnegie–Mellon Univ., 1988.

[2] J. Diederich, Knowledge representation and learning in a structured neural network. In: *Proceedings, Workshop Konnektionismus*, TR 329, GMD St. Augustin, 1988. Pp. 47–62.

[3] C. P. Dolan and P. Smolensky, Implementing a Connectionist Production System Using Tensor Products. TR UCLA-AI-88-15, Univ. of California, Los Angeles, 1988.

[4] M. G. Dyer, Symbolic NeuroEngineering for Natural Language Processing: A Multilevel Research Approach. TR UCLA-AI-88-14, Univ. of California, Los Angeles, 1988.

[5] W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1, 3rd Ed. J. Wiley & Sons, New York, N.Y., 1968.

[6] P. Floréen, P. Myllymäki, P. Orponen, and H. Tirri, Neural representation of concepts for robust inference. To appear in: *Proceedings, Computational Intelligence II, Milano, Sept. 1989.*

[7] J. A. Hendler, Marker-passing over microfeatures: towards a hybrid symbolic/connectionist model. *Cognitive Science* 13 (1989), 79–106.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, Ca., 1988.

[9] L. Shastri, *Semantic Networks: An Evidential Formalization and Its Connectionist Realization.* Pitman, London, 1988.