Report Number: WUCS-91-52

1991-12-01

Representation and Learning of Propositional Knowledge in Symmetric Connectionist Networks

Gadi Pinkas

The goal of this article is to construct a connectionist inference engine that is capable of representing and learning nonmotonic knowledge. An extended version of propositional calculus is developed and is demonstrated to be useful for nonmonotonic reasoning and for coping with inconsistency that may be a result of noisy, unreliable sources of knowledge. Formulas of the extended calculus (called penalty logic) are proved to be equivalent in a very strong sense to symmetric networks (like Hopfield networks and Boltzmann machines), and efficient algorithms are given for translating back and forth between the two forms of knowledge representation. The... **Read complete abstract on page 2.**

Representation and Learning of Propositional Knowledge in Symmetric Connectionist Networks

Gadi Pinkas

Complete Abstract:

The goal of this article is to construct a connectionist inference engine that is capable of representing and learning nonmotonic knowledge. An extended version of propositional calculus is developed and is demonstrated to be useful for nonmonotonic reasoning and for coping with inconsistency that may be a result of noisy, unreliable sources of knowledge. Formulas of the extended calculus (called penalty logic) are proved to be equivalent in a very strong sense to symmetric networks (like Hopfield networks and Boltzmann machines), and efficient algorithms are given for translating back and forth between the two forms of knowledge representation. The paper presents a fast learning procedure that allows symmetric networks to learn representations of unknown logic formulas by looking at examples. A connectionist inference engine is then sketched whose knowledge is either compiled from a symbolic representation or that is inductively learned from training examples. Finally, the paper shows that penalty logic can be used as an high-level specification language for connectionist networks, and as a framework into which several recent systems may be mapped.

Representation and Learning of Propositional Knowledge in Symmetric Connectionist Networks

Gadi Pinkas

WUCS-91-52

December, 1991

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Representation and Learning of Propositional Knowledge in Symmetric Connectionist Networks

Gadi Pinkas *
December 12, 1991
WUCS-91-52

Computer Science Dept.,
Campus Box 1045,
Washington University,
St. Louis, MO 63130
E.mail: pinkas@cics.wustl.edu. Tel:(314) 935-7526

Abstract

The goal of this article is to construct a connectionist inference engine that is capable of representing and learning nonmonotonic knowledge. An extended version of propositional calculus is developed and is demonstrated to be useful for nonmonotonic reasoning and for coping with inconsistency that may be a result of noisy, unreliable sources of knowledge. Formulas of the extended calculus (called penalty logic) are proved to be equivalent in a very strong sense to symmetric networks (like Hopfield networks and Boltzmann machines), and efficient algorithms are given for translating back and forth between the two forms of knowledge representation. The paper presents a fast learning procedure that allows symmetric networks to learn representations of unknown logic formulas by looking at examples. A connectionist inference engine is then sketched whose knowledge is either compiled from a symbolic representation or that is inductively learned from training examples. Finally, the paper shows that penalty logic can be used as an high-level specification language for connectionist networks, and as a framework into which several recent symbolic systems may be mapped.

*This research was supported by NSF grant 22-1321 57136.

1. Introduction

Humans seem to be able to reason about the surrounding world from a noisy and incomplete knowledge and with remarkably high speed. They are astoundingly good at inferring useful and often reliable information even from knowledge that is self-contradicting and sometimes erroneous.

It has been a decade now that AI has realized that the analysis of such reasoning mechanisms is a major task. As a result, many nonmonotonic systems have been proposed as formal models for this kind of reasoning. Some well known examples are circumscription [McCarthy 80] and default logic [Reiter 80].

Research in non-monotonic reasoning has tried to understand the basic mechanisms and the rationale behind our intuition when dealing with incomplete description of the world. Recent nonmonotonic systems are quite successful in capturing our intuitions (for examples see [Geffner 89], [Simari, Loui 90]). Most systems, however, are still plagued with intractable computational complexity, sensitivity to noise, inability to combine other sources of knowledge (like probabilities, utilities,...) and inflexibility to adjust themselves to new situations, and to develop personal intuitions.

Connectionist systems may be the missing link. They can supply us with a fast, massively parallel platform, and their ability to learn may be used to incorporate new data and dynamically change the knowledge base.

While scientists in traditional, symbolic AI were concentrating on development of powerful knowledge representation systems, connectionists were concentrating on powerful learning and adaptation mechanisms. Connectionism was criticized for lacking mechanisms like compositionality and systematicity, which are essential for high-level cognitive tasks and are easy for symbolic approaches [Fodor, Pylyshyn 88]. We would like to have systems that have sufficient expressive power, that perform fast (the brain suggests massive parallelism) and that are capable of learning and adjusting. "Clearly, the ultimate goal for both scientific approaches is to find efficient learning procedures for representationally powerful systems" [Hinton 90]. This article studies networks that can represent and learn unrestricted propositional[1] rules.

[1]The approach can be expanded to represent unrestricted predicate logic [Pinkas 91d].

One big difference between connectionist networks and symbolic knowledge representations is that symbolic systems need an interpreter to process the information expressed in the representation, and to reason with it. Connectionist networks have no such interpreter. The interpreter and the control mechanism should be included in the knowledge that is being represented. We strive therefore to find a connectionist representation that is capable of representing the information, as well as the procedural knowledge that is needed for control.

Among the different connectionist models, I choose to consider those with a symmetric matrix of weights. This family of models includes Hopfield networks [Hopfield 82a], [Hopfield 84b], Boltzmann machines [Hinton, Sejnowski 86], harmony theory [Smolensky 86], mean field theory [Hinton 89], and other variations. The reasons for using *symmetric* connectionist networks (SCNs) are the following:

1. symmetric networks can be characterized by energy functions which make it easier to specify the networks' behavior [Feldman 85];
2. symmetric networks were used successfully to express and solve (approximate) "hard" problems [Hopfield, Tank 85];
3. symmetric networks are capable of representing a large set of asymmetric networks [Pinkas 91e];[2] therefore they are quite powerful, and we will not lose expressive power if we restrict ourselves to the symmetric case.[3]

Ideally, we would like a wide range of logical formalisms to be representable in connectionist networks; however, we will be satisfied even if only some but general nonmonotonic frameworks will be represented. Also, it would be beneficial if we had a formal, declarative language that is capable of describing the knowledge encapsulated in a network. Such high-level, declarative language may then be used for specification and "programming" of connectionist networks. In the spirit of implementational connectionism, such language may be used as an intermediate level of abstraction between high-level cognitive processes and their low-level connectionist implementation.

[2] In fact, every non-oscillating network of binary threshold units is representable in SCNs.

[3] Sometimes an asymmetric form of a symmetric network will perform better; therefore, for efficiency, we may consider not to restrict ourselves to the symmetric case.

To summarize, my long term goal is to have an inference mechanism that 1) is highly expressive and yet capable of learning and adjusting; 2) is capable of nonmonotonic reasoning and of coping with inconsistent and noisy knowledge; 3) is fast on the average and capable of intelligent guessing when facing intractable problems (trades time with accuracy); 4) is capable of representing both data and control information; 5) is useful as a framework for the mapping of many symbolic systems.

My purpose in this article is to show that 1) nonmonotonic knowledge can be captured naturally in SCNs; 2) knowledge that is encapsulated in any SCN can be described by an extended version of propositional logic; 3) SCNs can learn representations of unknown formulas by looking at examples of truth assignments that satisfy the formulas; and finally, 4) SCNs can used as inference mechanisms that are capable of capturing both the information embedded in the logic formulas as well as the procedural knowledge used for control.

The paper is organized in the following way: section 2 presents penalty logic, its semantics and its proof-theory. The section demonstrates the usefulness of the new logic for nonmonotonic reasoning. In section 3, symmetric connectionist networks are introduced and the energy paradigm is reviewed. Section 4 defines equivalence of forms of knowledge representation and proves a strong equivalence between penalty logic formulas and SCNs. Section 5 discusses a learning algorithm that enables SCNs to learn representations of unknown propositional formulas inductively. An inference engine is sketched in section 6. Section 7 discusses related work, and section 8 concludes.

2. Penalty Logic

I'll extend now propositional calculus so that it will be useful for nonmonotonic reasoning and for coping with inconsistency. Later, this calculus will be mapped into SCNs.

The extended calculus is capable of expressing strength of belief, reliability of sources of knowledge, etc., by adding a real positive number (penalty) to every belief. This penalty may be assigned a variety of interpretations, for example, the numbers may represent "certainties" or "likelihoods" as in [Shortliffe 76], priorities as in [Brewka 89],[Lifschitz 85] or maximal entropy constraints as in [Goldszmidt, et al. 90]. When the knowledge sources are unreliable, a penalty may represent a measure of reliability [Rescher, Manor 70]. Note that some of these systems compute the penalties from less explicit information, while other systems let the user specify the penalties explicitly. I do not insist on a particular use or interpretation of the penalties, since the intention is to develop a *general* framework into which many logicist systems could be reduced (possibly with a variety of interpretations).

2.1. Extending propositional calculus

DEFINITION **2.1** A *penalty logic well formed formula* (PLOFF) ψ is a finite set of pairs. Each pair is composed of a real positive number, called *penalty*, and a standard propositional formula, called *assumption* (or belief); i.e., $\psi = \{< \rho_i, \varphi_i >| \; \rho_i \in \mathcal{R}^+, \varphi_i \text{ is a WFF}, i = 1...n\}$.
The set of the beliefs that are in ψ (denoted by $\mathcal{U}_\psi$) is $\mathcal{U}_\psi = \{\varphi_i \,|< \rho_i, \varphi_i >\in \psi\}$.

EXAMPLE **2.1** The Nixon diamond can be stated as:

1000	$N \rightarrow R$	Nixon is a republican
1000	$N \rightarrow Q$	Nixon is also a quaker
10	$R \rightarrow \neg P$	Republicans tend to be not pacifist
10	$Q \rightarrow P$	Quakers tend to be pacifist
3000	N	The person we reason about is Nixon.

An illustration of the example is shown in figure 1.

The set of the beliefs in the example is inconsistent; however, the penalties in this example reflect the strength with which we believe each proposition. High penalty is given to strict logic rules (facts), like the one that states that Nixon is a republican. We cannot allow strict facts to be defeated. The last fact (N) states that Nixon is the one we reason about. This fact receives the highest penalty of all since it is considered as *evidence*. The evidence is not usually part of our knowledge base and we would like to "jump" to conclusions once it is given. The evidence in this case is considered a temporary (corrigible) but very certain (infallible). Lower penalties are given to "defeasible" rules ("tend to be" rules), like the one that states that republicans tend to be not pacifist. When we know that somebody is a republican, we tend to believe that the person is not pacifist (by default); however, this "jumping" to conclusion

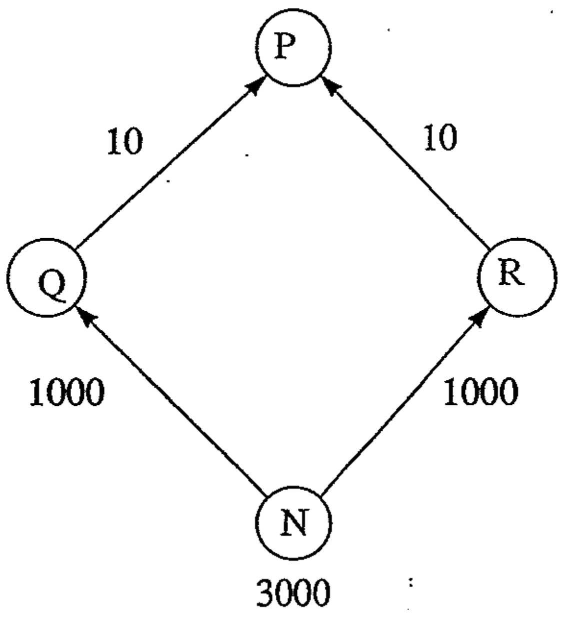

Figure 1: An illustration of the Nixon diamond as an inheritance network: nodes represent atomic propositions; the numbers are the penalties.

is blocked, if we know that the person is an exception to the rule. For example, we wouldn't like to conclude that a quaker is pacifist, if the person is also a republican, or if it was explicitly mentioned that the person is not pacifist. Clearly, we would like to conclude that Nixon is both a republican (R) and a quaker (Q); however, we would not like to conclude anything about the pacifism of Nixon. We don't have adequate reason either to believe P nor $\neg P$; therefore P is considered ambiguous.

If we would like to express the belief that religious ideas are stronger than political affiliations in influencing one's pacifism, then we may increase the penalty for $Q \rightarrow P$ to 15; leaving the penalty for $R \rightarrow \neg P$ unchanged (10).

EXAMPLE 2.2

1000 $N \rightarrow R$	Nixon is a Republican
1000 $N \rightarrow Q$	Nixon is also a Quaker
10 $R \rightarrow \neg P$	Republican tend to be not pacifist
15 $Q \rightarrow P$	Quakers tend to be pacifist
3000 N	The person we reason about is Nixon.

In the revised set of assumptions, we have two competing arguments. One argument supports the pacifism of Nixon while the other supports its negation. The pacifism of Nixon is not ambiguous in this case, since the argument that supports P wins (the winning argument is stronger and therefore manages to defeat the disagreeing argument [Loui 87]).

2.2. Model-theory

There are many ways to interpret the penalties and the assumptions in our formalism. I shall give one such interpretation that is convenient, useful and general.

Given a knowledge base $\psi = \{< \rho_i, \varphi_i >\}$, the PLOFF ψ determines a ranking over the set of all possible models (truth assignments of n atomic propositions). This ranking reflects "normality" or "goodness" we tend to associate with possible models of the world (see [Shoham 88]). By specifying ψ we mean informally that models that satisfy many "important" assumptions are "better" than models that satisfy fewer or less important assumptions. Every two models may always be compared by looking at the assumptions (in ψ) that are violated. Two models that violate the same set of assumptions are considered to be "equally good". Even if the models violate different sets of assumptions but the sum of the penalties of both sets is the same, then the two models are "equally good". A model is more "normal" (or "better") than another model if the sum of the penalties of the violated assumptions of the first is less than the sum of the penalties violated by the second.

This interpretation of the penalties induces a ranking function that assigns a real value (rank) to all the possible models. The ranking function that is induced is called the violation rank of ψ:

DEFINITION 2.2 The *violation-rank* of a PLOFF ψ is the function ($Vrank_\psi$) that assigns a real-valued rank to each of the truth assignments. The $Vrank_\psi$ for a truth assignment $\vec{x}$ is computed by summing the penalties for the assumptions of ψ that are violated by the assignment; i.e., $Vrank_\psi(\vec{x}) = \sum_{\vec{x} \not\models \varphi_i} \rho_i$.

DEFINITION 2.3 The models that minimize the $Vrank_\psi$ function are called the *preferred models* of ψ; i.e., $\{\vec{x} \mid min_{\vec{y}}\{Vrank_\psi(\vec{y})\} = Vrank_\psi(\vec{x})\}$. The set of all preferred models is denoted by Γ_ψ.

DEFINITION **2.4** Let φ, ψ be PLOFFs, a PLOFF ψ ***semantically entails*** φ $(\psi \models \varphi)$ iff all the preferred models of ψ are also the preferred models of φ; i.e., $\Gamma_\psi \subseteq \Gamma_\varphi$.

Note that a sentence ψ therefore entails φ iff any model that minimizes the violation-rank of ψ, also minimizes the violation-rank of φ.

In the Nixon example, the preferred models are only two: $(NRQP)$ and $(NRQ\neg P)$. Examples of some valid conclusions are therefore $N, R \wedge Q$, etc. (since these conclusions are satisfied by all the preferred models). The pacifism of Nixon is ambiguous, since P holds in one preferred model while $\neg P$ holds in the other.

2.3. Merging PLOFFs, evidence and background knowledge

The operator *merge* ($\cup$) in the meta language, plays the role of $\wedge$ (AND) in classic propositional logic. It allows us to combine two PLOFFs simply by merging them.

DEFINITION **2.5** The *merge* operation ($\overset{*}{\cup}$) is defined: $\psi_1 \overset{*}{\cup} \psi_2 = (\psi_1 - \psi_2) \cup (\psi_2 - \psi_1) \cup \{< 2\rho_i, \varphi_i > | < \rho_i, \varphi_i > \in \psi_1 \cap \psi_2\}$.

The reader may check that $Vrank_{\psi \cup \psi'} = Vrank_\psi + Vrank_{\psi'}$ (when $\psi \cap \psi' = \phi$).

The merge operation allows us an incremental update of the knowledge. Combining new rules with existing ones is simply done by adding the appropriate $Vrank$ functions. Later, after the equivalence of networks and logic formulas is established, we'll see that this property allows to add or delete a PLOFF from an existing network only by adding or deleting the relevant energy terms. There is no need to re-compute the new network all over again when some updates occur.

Nonmonotonic systems "jump" to conclusions based on evidence given, and later may retract those conclusions based on new evidence. It is therefore convenient to divide the knowledge from which we want to reason into "background" knowledge and "evidence" [Geffner 89], [Poole 85]. The background knowledge is relatively fixed, and there should be an easy way to combine evidence with it. In our formalism, combining the evidence is simply done by merging the evidence and the background or adding together the two ranking functions.

DEFINITION **2.6** Let ψ, e, φ be PLOFFs. Evidence e entails φ with respect to a background knowledge ψ ($e \overset{\psi}{\models} \varphi$), iff $\psi \overset{*}{\cup} e \models \varphi$. The *consequence relation* induced by ψ is the set of all pairs $< e, \varphi >$ such that $e \overset{\psi}{\models} \varphi$.

One special case of this definition is when the evidence is strict; i.e., its validity is certain. This special case is very useful, and indeed, in most reasoning systems the evidence is never defeated, and the agent draws conclusions based on the absolute validity of the evidence (e.g., [Geffner 89]). To represent a strict formula in penalty logic, the set $\mathcal{U}_\psi$ should be consistent and the penalties that are assigned to the assumptions should be higher than any other combination of background beliefs (the penalties are practically infinite).

DEFINITION **2.7** Strict evidence is a PLOFF $e = \{< \infty, e_i >\}$ such that the set $\mathcal{U}_e = \{e_i\}$ is consistent and ∞ represents a large penalty (larger than any combination of background beliefs).

EXAMPLE **2.3** In the Nixon diamond the following beliefs are considered background:

1000	$N \rightarrow R$	Nixon is a Republican.
1000	$N \rightarrow Q$	Nixon is also a Quaker.
10	$R \rightarrow \neg P$	Republicans tend to be not pacifist.
10	$Q \rightarrow P$	Quakers tend to be pacifist.

The fact $< 3000, N >$ is strict evidence that triggers for example the conclusion of $Q \wedge R$. Another example of strict evidence is $< 3000, \neg Q >$ that triggers the conclusion $\neg N$.

The nonmonotonicity of penalty logic is reflected by the property that sometimes conclusions need to be retracted when new evidence is added. In the example, when there is an evidence that some one is a Quaker the conclusion is that he or she is also pacifist. If in addition, we add the evidence that Nixon is that someone, we need to retract the conclusion.

Loyal to the goal of being as general as possible, I'll not restrict the evidence to being strict. Such generalization is not mere formality, and has its direct uses. Defeasible evidence is a phenomenon encountered in many practical applications. For example, when the evidence is obtained via sensory devices that are unreliable, redundant and noisy, our agent may "not believe its own eyes" if the evidence contradicts with some highly reliable facts of the background knowledge.

As with evidence, conclusions need not be strict. Most symbolic systems treat a conclusion as a strict proposition that either follows from the background knowledge, or its negation follows, or is ambiguous. Penalty logic allows conclusions to be stated as PLOFFs (with penalties). Thus, such conclusions may arrive for example as queries via noisy channels. The query we wish to prove may therefore be redundant and unreliable exactly as the evidence and the background knowledge.

2.4. Proof-theory

A sound and complete proof-theory can be shown for penalty logic. This proof-theory is based solely on syntactic considerations, and gives a clarifying look at the reasoning process in penalty logic.

Instead of ranking the models and using the "best" models for the reasoning process, we can rank consistent-subsets of the assumptions of ψ, and use the "best" (preferred) consistent-subsets to perform deduction. A conclusion is made in the proof-theory iff all the preferred consistent-subsets entail it.

DEFINITION **2.8** A set T is called a *theory* of a PLOFF ψ iff T is a *consistent* subset of the assumptions in ψ; i.e., the set of $T \subseteq \mathcal{U}_\psi$ is a subset of the assumptions in ψ and there is a model that satisfy T.

DEFINITION **2.9** The *penalty-function* of a theory T of ψ is the function obtained by summing the penalties of the assumptions in ψ that are not included in T; i.e., $penalty_\psi(T) = \sum_{\varphi_i \in (\mathcal{U}_\psi - T)} \rho_i$.

A ranking is therefore induced by ψ over the set of theories of ψ. This ranking is computed by summing the penalties of the missing assumptions.

DEFINITION **2.10** A *preferred* theory of ψ is a theory T that minimizes the penalty function of ψ; i.e., The set of the preferred theories of ψ is $T_\psi = \{T \mid penalty_\psi(T) = min_S\{penalty_\psi(S) \mid S \text{ is a theory of } \psi\}\}$.

DEFINITION **2.11** Let ψ, φ be PLOFFs, let $T_\psi = \{T_i\}$ the set of all preferred theories of ψ, and let $T_\varphi = \{T'_i\}$ the set of all preferred theories of φ.
We say the ψ ***entails*** [4] φ (denoted by $\psi \vdash \varphi$) iff all the preferred theories T_i of ψ entail (in the classical sense) the disjunction of all the preferred theories of φ; i.e.,

$$\psi \vdash \varphi \text{ iff } \bigvee_{T_i \in T_\psi} T_i \vdash \bigvee_{T'_j \in T_\varphi} T'_j$$

As a special case, consider the case where the conclusion φ is strict (φ is a classical propositional formula without penalties). A PLOFF ψ entails φ iff every preferred theory of ψ entails φ in the classical sense of entailment.

In the Nixon example, the set of assumptions is inconsistent (leads to a contradiction); however, there are $2^5 - 2$ non-empty consistent subsets where at least one belief of ψ is missing. If we rank each of the consistent subsets by summing the penalties of the missing beliefs, we get that the preferred theories are $T_1 = \{N, N \rightarrow Q, N \rightarrow R, Q \rightarrow P\}$ and $T_2 = \{N, N \rightarrow Q, N \rightarrow R, R \rightarrow \neg P\}$. These preferred theories are each ranked 10 since only one belief in ψ (of strength 10) is missing in each such theory.

Each of the two preferred theories entails the obvious conclusions (like $N, Q \wedge R$), but neither P nor $\neg P$ can be concluded, since the two preferred theories do not agree on either. The reasoning process can be intuitively understood as a competition among consistent subsets. The subsets that win are those theories with minimal penalty. A conclusion is entailed only if all the winners conclude it independently.

We'll need the next two lemmas to show that the proof-theory is sound and complete.

Lemma 1 *Let $T \subseteq \mathcal{U}_\psi$ be a consistent subset of the assumptions in ψ.*
The subset T is maximal-consistent[5] *in ψ iff every model that satisfies T has a violation-rank equal to the penalty of T; i.e., T is a maximal-consistent subset iff $(\forall \vec{x})$ if $\vec{x} \models T$ then $penalty_\psi(T) = Vrank_\psi(\vec{x})$.*

Proof: If T is a maximal-consistent subset of ψ, the assumptions in ψ that are left out of T are also the assumptions that are violated by any model $\vec{x}$ that satisfies T (otherwise such assumptions are consistent with

[4] Note that the deductive closure of preferred theories roughly resemble extensions (like in [Reiter 80]). The definition of entailment in penalty logic resembles therefore entailment by intersection of all extensions.
[5] A subset T is maximal-consistent if no other assumption of ψ can be added to T while still preserving the consistency of the set.

T and therefore T is not maximal). Also, every assumption that is violated by a model $\vec{x}$ that satisfies T cannot be in T. Therefore, if T is a maximal-consistent subset then for every model $\vec{x}$ of T, the set of missing assumptions in T is equal to the set of assumptions violated by $\vec{x}$. Therefore $penalty_\psi(T) = Vrank_\psi(\vec{x})$.

Assume that every model $\vec{x}$ that satisfies T has $Vrank_\psi(\vec{x}) = penalty_\psi(T)$. If T is not maximal-consistent then there is an assumption in ψ that can be included into T and have a model $\vec{x}$ satisfying both T and the new assumption. The violation rank of $\vec{x}$ must be lower than the penalty of T since the set of assumption not included in T subsumes the set of assumptions violated by $\vec{x}$ and contains at least one assumption not violated by $\vec{x}$; i.e., $Vrank_\psi(\vec{x}) < penalty_\psi(T)$. This is a contradiction with the assumption that $penalty_\psi(T) = Vrank_\psi(\vec{x})$.

□

The reader may observe that any preferred-theory of ψ is a maximal-consistent subset of $\mathcal{U}_\psi$ and therefore the penalty of a preferred theory is equal to the violation rank of its satisfying models. This allows us to use a proof-theoretic ranking function ($penalty_\psi$) instead of the model-theoretic function ($Vrank_\psi$).

The next lemma establishes the relationship between preferred models and preferred theories.

Lemma 2 *A model $\vec{x}$ is a preferred model of a PLOFF ψ iff the model $\vec{x}$ satisfies some preferred theory of ψ.*

Proof: If $\vec{x}$ is a preferred model of ψ then it minimizes $Vrank_\psi$. Let T be the set composed of all assumptions in ψ that are satisfied by $\vec{x}$. Since the assumptions that are violated by $\vec{x}$ are exactly those that are not included in T, we deduce that $Vrank_\psi(\vec{x}) = penalty_\psi(T)$. But if T is not a preferred theory then there exists a preferred theory T' such that $penalty_\psi(T') < penalty_\psi(T)$. By lemma 1, the models $\vec{y}$ that satisfy T' have $Vrank_\psi(\vec{y}) = penalty_\psi(T')$, and we conclude that $Vrank_\psi(\vec{y}) = penalty_\psi(T') < penalty_\psi(T) = Vrank_\psi(\vec{x})$. This is a contradiction to the minimality of $Vrank(\vec{x})$.

If $\vec{x}$ is a model of a preferred theory T of ψ then T minimizes the penalty function. By lemma 1, $Vrank_\psi(\vec{x}) = penalty_\psi(T)$. If $\vec{x}$ is not a preferred model of ψ then there must be a preferred model $\vec{y}$ such that $Vrank_\psi(\vec{y}) < Vrank_\psi(\vec{x})$. Let T' the set of all assumptions of ψ satisfied by $\vec{y}$. The set T' has $penalty_\psi(T') = Vrank_\psi(\vec{y})$, since the set of the assumptions violated by $\vec{y}$ is equal to the set of assumptions not included in T'. Therefore,

$penalty_\psi(T') = Vrank_\psi(\vec{y}) < Vrank_\psi(\vec{x}) = penalty_\psi(T)$, in contradiction to the minimality of $penalty_\psi(T)$.

□

Theorem 1 *The proof procedure is sound and complete; i.e., $\psi \models \varphi$ iff $\psi \vdash \varphi$.*

Proof: If $\psi \models \varphi$ then every preferred model of ψ is also a preferred model of φ. Based on lemma 2, every preferred model of ψ satisfies some preferred theory T of ψ and also satisfies some preferred theory of φ. Therefore, every preferred model of ψ satisfies the disjunction of the preferred theories of φ. From lemma 2 every model that satisfies a preferred theory T of ψ is also a preferred model of ψ and therefore satisfies the disjunction of the preferred theories of φ; i.e., $T \vdash \bigvee_{T'_j \in T_\varphi} T'_j$. We conclude therefor that $\psi \vdash \varphi$.

If $\psi \vdash \varphi$ then every model that satisfies a preferred theory T of ψ also satisfies a preferred theory T' of φ. From lemma 2, a model that satisfies T' is also a preferred model of φ and therefore, every model that satisfies T is also a preferred model of φ. Based on lemma 2 every preferred model T of ψ satisfies some preferred theory of ψ and therefore is a preferred model of φ ($\Gamma_\psi \subseteq \Gamma_\varphi$). We therefore conclude that $\psi \models \varphi$.

□

This sound and complete proof mechanism of competing theories is useful for both dealing with inconsistency in the knowledge base and for defeasible reasoning. For example, when we detect inconsistency, we usually want to adopt a theory with maximum cardinality (we assume that only a minority of the observations are erroneous). Indeed, in penalty logic, when all the penalties are one, the theories that win have maximal cardinality and only a minority of the assumptions is defeated. Thus, minimum penalty means maximum cardinality. Penalty logic is therefore a generalization of the maximal cardinality principle which is useful when coping with noisy and inconsistent sets of beliefs. For defeasible reasoning, the notion of conflicting theories can be used to decide between conflicting sets of arguments. Intuitively, a set of arguments A_1 defeats a conflicting set of arguments A_2 if A_1 is supported by a "better" theory than all those that support A_2 (see [Loui 87] and [Simari, Loui 90] for a discussion on argument systems).

EXAMPLE 2.4 Two levels of blocking (from [Brewka 89]):

1	meeting	I usually go to the Monday meeting.
10	sick $\rightarrow$($\neg$ meeting)	If I'm sick I usually don't go to the meeting.
100	cold-only $\rightarrow$ meeting	If I have only a cold then I tend go to the meeting.
1000	cold-only $\rightarrow$ sick	If I have a cold it means I'm sick.

Without any additional evidence, all the assumptions are consistent, and we can infer that "meeting" is true (from the first assumption). However, given the evidence that "sick" is true, we prefer theories that falsify "meeting" and "cold-only", since the second assumption has greater penalty than the competing first assumption (the only theory that wins does not include the first assumption). If we include the evidence "cold-only" then the theory that previously won loses now, and the new winner is the theory that does not include the second assumption. As a result, the conclusion "meeting" is drawn despite the fact that "sick" is also concluded.

3. Energy functions

This section reviews[6] symmetric connectionist models and the energy minimization paradigm. Later, we'll show the relationship between penalty logic and the energy paradigms.

Searching for a global minima for quadratic functions is the essence of symmetric connectionist models used for parallel constraint satisfaction. Examples for such models are Hopfield networks, Boltzmann machines, harmony theory and mean-field theory. These models are characterized by a recurrent network architecture, a symmetric matrix of weights (with zero diagonal) and a quadratic energy function that should be minimized. Each unit asynchronously computes the gradient of the function and adjusts its activation value, so that energy decreases gradually. The network eventually reaches equilibrium, settling on either a local or a global minimum. [Hopfield 84b] demonstrated that certain complex optimization problems can be stated as constraints that are expressed in quadratic energy functions and be approximated using these kind of networks.

There is a direct mapping between these networks and the quadratic energy functions they minimize. Every quadratic energy function can be translated into a corresponding network and vice versa. Weighted arcs (i.e., pairwise connections) in the network correspond to weighted terms of two variables in the energy function (with opposite sign). Thresholds of units in the network correspond to single-variable terms in the function. Most of the time we will not distinguish between the function and the network that minimizes it. An example of a network and its energy function is given in figure 2.

[6]The figures are taken from [Pinkas 90b].

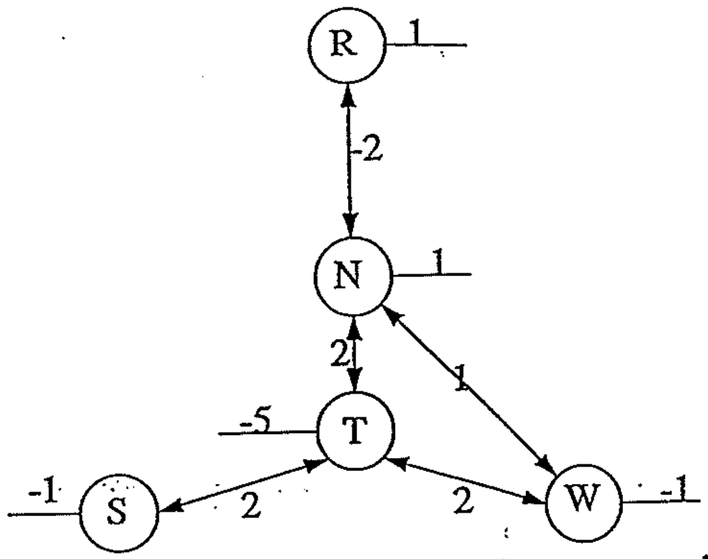

Figure 2: A symmetric network that represents the function $E = -2NT - 2ST - 2WT + 5T + NS + RN - WN + W$.

3.1. High-order energy functions

To represent arbitrary logic formulas, a network will need the power of either high-order connections or hidden units. This section reviews high-order networks, and shows how to convert them into standard (pairwise) networks by introducing new hidden units.

High-order connectionist networks have sigma-pi units [Rumelhart et al. 86] with multiplicative connections. Symmetric networks can be easily extended to handle high-order connections. Naturally, such networks may be viewed as minimizing high-order energy functions [Sejnowski 86].

A k-order energy function is a function $E : \{0,1\}^n \rightarrow \mathcal{R}$ that can be expressed as sum of products, with product terms of up to k variables. A k-order energy function is denoted by: $E^k(x_1, \ldots, x_n) =$

$$\sum_{1 \le i_1 < i_2 < \cdots < i_k \le n} w_{i_1,\ldots,i_k} X_{i_1} \cdots X_{i_k} + \sum_{1 \le i_1 < \cdots < i_{k-1} \le n} w_{i_1,\ldots,i_{k-1}} X_{i_1} \cdots X_{i_{k-1}} + \cdots + \sum_{1 \le i \le n}^{1} w_i X_i$$

Quadratic energy functions (or second-order functions) are special cases of the high-order case:

$$\sum_{1 \le i < j \le n} w_{ij} X_i X_j + \sum_{i \le n} w_i X_i.$$

In the high-order model each node is assigned a sigma-pi unit that updates its activation value using:

$$net_i = \frac{dE}{dX_i} = \sum_{i_1 \cdots i \cdots i_k} -w_{i_1,\ldots,i,\ldots,i_k} \prod_{1 \leq j \leq k, i_j \neq i} X_{i_j}$$

$$a_i = F(net_i)$$

where $a_i = F(net_i)$ is the standard update rule that is unique to the model we wish to extend. In the Hopfield model for example, $F(net_i) = 1$ if $net_i > 0$, and $F(net_i) = 0$ otherwise. A high-order network (see figure 3) is a hyper graph, where k-order terms are translated into hyper-arcs connecting k nodes. The arcs are not directed (the weight is the same for every node that is part of the arc) and the weight of an arc is determined by the weight of the corresponding term in the energy function (with an opposite sign).

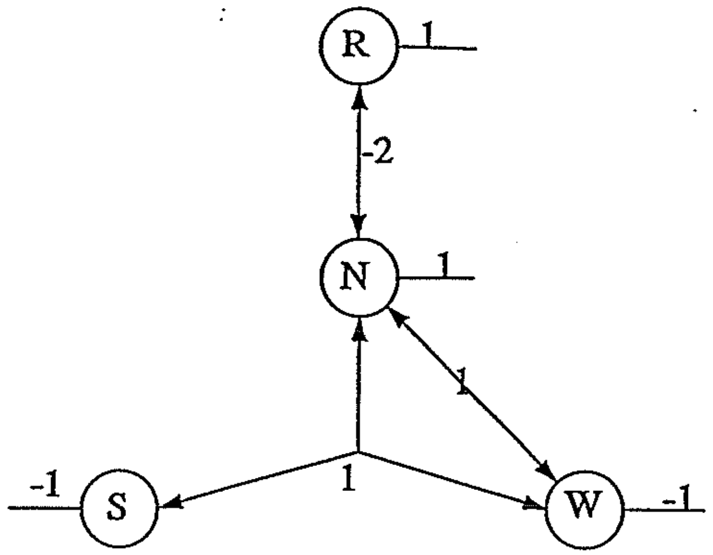

Figure 3: A cubic network that represents $E = -NSW + NS + RN - WN + W$ using sigma-pi units and a cubic hyper-arc (it is equivalent to the network of figure 1 without the hidden unit T).

As in the quadratic case, there is a translation back and forth between k-order energy functions and symmetric high-order networks with k-order sigma-pi units. We can arbitrarily divide the variables of an energy function into two sets: visible variables and hidden variables. The hidden variables correspond to the hidden units of the network, and the visible variables correspond to the visible units. An energy function with both hidden and visible variables is denoted usually as a function $E(\vec{x},\vec{t})$, where $\vec{x}$ represents the visible variables and $\vec{t}$ represents the hidden variables.

An assignment of zeros and ones to the visible variables is called a visible state. The values of the visible units after an equilibrium was reached, are considered as the "answer" of the network.

Later in this article, I'll interpret visible states as truth assignments: the visible variables are viewed as atomic propositions: "1" is interpreted as "true" and "0" is interpreted as "false".

We call the set of minimizing vectors projected onto the visible variables, "the visible solutions" of the minimization problem; i.e., $\{\bar{x} \mid (\exists \bar{t}) E(\bar{x},\bar{t}) = min_{\bar{y},\bar{z}}\{E(\bar{y},\bar{z})\}\}$. Models like Boltzmann machines, harmony theory, mean field theory, as well as many other variations, may be viewed as searching for a *global minimum*[7] of the corresponding energy functions. Local minima or spurious memories may exist. In general however, local minima are considered to be undesirable phenomena, and cause a degradation in the performance of the network. This article will ignore local minima that are not global, and usually they will not represent any meaningful knowledge.

DEFINITION **3.1** Let E be a symmetric network with energy function $E(\vec{x},\vec{t})$, where $\vec{t}$ designates the hidden variables. The *characteristic* function of the network is the function: $Erank_E(\vec{x}) = min_{\vec{y}}\{E(\vec{x},\vec{y})\}$.

The $Erank_E$ function defines the energy of all visible states. The energy of a visible state is the energy level obtained when the visible units are clamped with the state's values, and the hidden units are free to settle so that a minimum is reached. This $Erank_E$ function characterizes the network's behavior: it is independent of the hidden units and it is also independent of the exact topology of the original network. There may be many possible networks with the same characteristic function. The next section uses the characteristic function to show equivalence between different networks.

3.2. The equivalence between high-order networks and low-order networks

The following subsection is a review of results reported in [Pinkas 90b].

[7] Several global minima may exist, all with the same energy level.

We call two energy functions ***strongly equivalent***, if their corresponding characteristic ($Erank$) functions are equal up to a constant difference; i.e: $E_1 \approx E_2$ iff $Erank_{E_1} = Erank_{E_2} + c$. Networks that are strongly equivalent not only have the same set of global minima, but also have a very similar energy surface and induce the same ordering on the visible states; i.e., if s_1 and s_2 are visible states then "same ordering" means that $E_1(s_1) < E_1(s_2)$ iff $E_2(s_1) < E_2(s_2)$.

I'll show now an algorithm to convert any high-order network into a strongly equivalent low-order one with additional hidden units. In addition, any energy function with hidden variables can be converted into a strongly equivalent, (possibly) higher-order network by eliminating some or all of the hidden units. These algorithms allow us to trade the computational power of sigma-pi units for additional simple units and vice versa. As a result we'll see that the expressive power of high-order networks is the same as that of low-order networks with hidden units.

Readers who are not interested in the technical details of the constructions may skip now to the next subsection. They may keep in mind only that the constructions for both directions are possible and efficient.

Theorem 2

- *Any k-order term ($w\prod_{i=1}^{k} x_i$), with* NEGATIVE *coefficient w, can be replaced by the quadratic terms: $\sum_{i=1}^{k} 2wX_iT - (2k-1)wT$ generating a strongly equivalent energy function with one additional hidden variable T.*
- *Any k-order term ($w\prod_{i=1}^{k} x_i$), with* POSITIVE *coefficient w, can be replaced by the terms: $w\prod_{i=1}^{k-1} x_i - (\sum_{i=1}^{k-1} 2wX_iT) + 2wX_kT + (2k-3)wT$, generating a strongly equivalent energy function of order $k-1$ with one additional hidden variable T.*

Proof appears in [Pinkas 90a].

EXAMPLE 3.1 The following is a 4-order energy function with a 4-order term $XYZU$. It can be converted into a quadratic energy function using two additional hidden variables T and T'.

$$\begin{aligned}
-XY + XYZU &\approx -XY + XYZ - 2XT - 2YT - 2ZT + 2UT + 5T \\
&\approx -XY + XY - 2XT' - 2YT' + 2ZT' + 3T' - 2XT - 2YT - 2ZT + 2UT + 5T \\
&= -2XT' - 2YT' + 2ZT' + 3T' - 2XT - 2YT - 2ZT + 2UT + 5T
\end{aligned}$$

The symmetric transformation, from low-order into high-order functions by eliminating any subset of the variables, is also possible (of course we are interesting in eliminating only hidden variables).

To eliminate T, bring the energy function to the form: $E = E' + oldterm$, where $oldterm = (\sum_{j=1}^{k} w_j \prod_{i=1}^{l_j} X_{j_i})T$.

Consider all assignments S for the variables ($\hat{X} = x_{i_1} \cdots x_{i_l}$) in *oldterm* (not including T), such that $\beta_S = \sum_{j=1}^{k} w_j \prod_{i=1}^{l_j} x_{j_i} < 0$.

Let $L_S^j = \begin{cases} \text{“}X_{i_j}\text{”} & \text{if } S(X_{i_j}) = 1 \\ \text{“}(1 - X_{i_j})\text{”} & \text{if } S(X_{i_j}) = 0 \end{cases}$ it is the expression “X_i” or “$(1 - X_i)$” depending whether the variable is assigned 1 or 0 in S.

The expression $\prod_{j=1}^{l} L_S^j$ therefore determines the state S, and the expression

$$newterm = \sum_{\text{S such that } \beta_S < 0} \beta_S \prod_{j=1}^{l} L_S^j$$

represents the disjunction of all the states that cause a reduction in the total energy.

The new function $E' + newterm$, is therefore equivalent to $E' + oldterm$ and does not include T.

EXAMPLE 3.2 Let T be the hidden variable to be eliminated, then:

$$AB + TAC - TA + 2TB - T = AB + T(AC - A + 2B - 1)$$

The following assignments for (A, B, C) cause β to be less then zero:

$\beta_{(0,0,0)} = -1$

$\beta_{(0,0,1)} = -1$

$\beta_{(1,0,0)} = -2$

$\beta_{(1,0,1)} = -1$

The new term equals:

$$-(1-A)(1-B)(1-C) - (1-A)(1-B)C - 2A(1-B)(1-C) - A(1-B)C = -ABC + AB + AC - A + B - 1$$

Therefore:

$$AB + TAC - TA + 2TB - T \approx -ABC + 2AB + AC - A + B.$$

4. The equivalence between penalty-logic and energy minimization

This section defines equivalence between different forms of knowledge representation, and use these definitions to show the relationships between penalty logic and SCNs.

4.1. Reasoning with ranking functions

A ranking function over a set of models is a function that assigns a real value (rank) to every model in the set. The ranking of the models may be considered as a grade for the "normality" or the "goodness" of the models.

As we saw in previous subsection, every SCN E is characterized by the ranking function $Erank_E$. Similarly, every ranking function is equal to some high-order energy function and therefore characterizes some SCN.[8] A search performed by the SCN for a global minimum may be viewed thus as a search for a model that minimizes the ranking function.

Penalty logic formulas, classical logic WFFs, and SCNs may be interpreted as representations of ranking functions. It may be useful therefore to define our reasoning mechanism independently of the knowledge representation form:

DEFINITION **4.1** Let $\mathcal{W} = \{0,1\}^n$ be the set of models defined over a set of n atomic propositions.
A *ranking function* $k : \mathcal{W} \rightarrow \mathcal{R}$ is a function that maps models into reals.
A ranking function k is *strict* iff the domain of k is $\{0, \infty\}$ (where ∞ represents a large positive number).
A *preferred model* $\vec{x}$ of a ranking function k is a model that minimizes k; i.e., $k(\vec{x}) = min_{\vec{y}}\{k(\vec{y})\}$.
The set of preferred models of k is denoted Γ_k.

DEFINITION **4.2** Let f, k, e ranking functions.
f is entailed from k ($k \models f$) iff $\Gamma_k \subseteq \Gamma_f$.
f is entailed from the background knowledge k using the evidence e ($e \overset{k}{\models} f$) iff $k + e \models f$.
The *consequence relation* induced by k is the set of all pairs $\{< e, f >| \; e \overset{\psi}{\models} f\}$.

[8]There is no guarantee that the the size of the network will be polynomial in n (the number of visible variables).

The reasoning mechanism defined for penalty logic in section 2 is consistent with the above definitions if $Vrank_{\psi}$ is taken as the ranking function.

4.2. Calculi to describe ranking functions

Our next step is to describe symbolically the knowledge that is encapsulated in a ranking function. This subsection defines several languages for describing of ranking function and shows their equivalence. Sentences of such languages are interpreted as ranking functions, and transformations are allowed from one knowledge representation into another if some basic properties are preserved.

The following definitions establish the relationship between a form of knowledge representation and its meaning.

DEFINITION 4.3 A calculus is a triple $< \mathcal{L}, m(), M >$, where $\mathcal{L}$ is a language, M is a set of possible models and $m : \mathcal{L} \rightarrow \{k \mid k \text{ is a ranking function }\}$ is a function that returns for each sentence of the language $\mathcal{L}$, a ranking function. $m(s)$ is called the *interpretation* of the sentence s.
Let $s, s', e, k \in \mathcal{L}$; a model $\vec{x}$ is a *preferred model* of s ($\vec{x} \models s$) iff $\vec{x}$ is a preferred model of the ranking function $m(s)$.
A sentence s *entails* sentence s' ($s \models s'$) if the ranking function $m(s)$ entails the ranking function $m(s')$.
Similarly, a combination of a background sentence with an evidence sentence is interpreted as the addition of their corresponding ranking functions; i.e., $e \overset{k}{\models} s$ iff $(m(e) + m(k)) \models m(s)$.
The consequence relation of k is the set of all pairs $\{< e, s > \mid e \overset{k}{\models} s\}$.

Both classic predicate logic and propositional logic can be viewed as calculi whose languages describe strict ranking functions.

EXAMPLE 4.1 Propositional calculus is $< \mathcal{L}, m(), \{0,1\}^n >$, where $\mathcal{L}$ is the language of propositional well formed formulae (WFFs) and $m(s)$ outputs the function $(\infty(1 - H_s(\vec{x})))$, given a formula s (∞ represents a large positive real). $H_s(\vec{X})$ is the characteristic function of the WFFs and is recursively defined as:

$$H_s(\vec{X}) = \begin{cases} X_i & \text{if } s = X_i \text{ is an atomic proposition} \\ 1 - H_{s'}(\vec{X}) & \text{if } s = \neg s' \\ H_{s_1}(\vec{X}) \times H_{s_2}(\vec{X}) & \text{if } s = s_1 \wedge s_2 \\ H_{s_1}(\vec{X}) + H_{s_2}(\vec{X}) - H_{s_1}(\vec{X}) \times H_{s_2}(\vec{X}) & \text{if } s = s_1 \vee s_2 \end{cases}$$

The reader may easily observe, that any propositional WFF describes a strict ranking function that returns 0 for truth assignments that satisfy the WFF, and ∞ for assignments that do not satisfy it.

EXAMPLE **4.2** Penalty logic is a calculus $< \mathcal{L}_p, m, \{0,1\}^n >$ such that $m(\psi) = Vrank_\psi$.

DEFINITION **4.4** Let $s \in \mathcal{L}_1$ and $s \in \mathcal{L}_2$ be sentences of two (possibly different) calculi $< \mathcal{L}_1, m, M >$ and $< \mathcal{L}', m', M >$; we define three kinds of equivalence relations between them:

1. s is *strongly equivalent* to s' ($s \overset{s}{\approx} s'$) iff their corresponding ranking functions are equal, up to a constant difference; i.e., $m(s) = m'(s') + c$. We call this equivalence "magnitude preserving" or s-equivalence.

2. s is p-equivalent to s' ($s \overset{p}{\approx} s'$) iff their associated ranking functions induce the same ordering over the set of models; i.e., $\forall \vec{x}, \vec{y}, m(s)(\vec{x}) < m(s)(\vec{y})$ iff $m'(s')(\vec{x}) < m'(s')(\vec{y})$. We call this equivalence "preference preserving" or p-equivalence.

3. s is weakly equivalent to s' ($s \overset{w}{\approx} s'$) iff their corresponding ranking functions have the same sets of satisfying models; i.e., $\Gamma_{m(s)} = \Gamma_{m'(s')}$. We call this equivalence "minima preserving" or w-equivalence.

OBSERVATION **4.1** *1. If two background sentences are strongly equivalent, then for any given evidence e, the two corresponding sentences entail the same set of conclusions; i.e., if $s \overset{s}{\approx} s'$ then for every evidence e and every conclusion c, $(m(s) + m(e)) \models m(c)$ iff $(m'(s') + m'(e)) \models m'(c)$. Therefore, two strongly equivalent sentences have the same induced consequence relation. i.e., In addition, the probabilistic meaning that is usually associated with the energy function of Boltzmann Machines is preserved, since $P(\vec{x})/P(\vec{y}) = e^{(m(s)(\vec{x}) - m(s)(\vec{y}))} = e^{((m'(s')(\vec{x})+c)-(m'(s')(\vec{y})+c))}$.*

2. *If two background sentences are p-equivalent, then for every* strict *evidence e, the two sentences entail the same set of conclusions; i.e., if* $dom(e) = \{0, \infty\}$ *and* $s \overset{p}{\approx} s'$, *then for every conclusion* c, $(m(s) + e) \models m(c)$ *iff* $(m'(s') + e) \models m'(c)$. *We can't guarantee this property for any non-strict evidence.*

3. *If two sentences* s, s' *are weakly equivalent, then the two sentences entail the same set of direct conclusions; i.e.,* $m(s) \models m(c)$ *iff* $m'(s') \models m'(c)$. *We can't guarantee this property to hold once we try to add evidence.*

The reader may easily observe that if two sentences are strongly equivalent then they are also p-equivalent and if they are p-equivalent they are also weakly equivalent.

If all we want is to preserve the set of conclusions achievable from a piece of knowledge, we may use transformations which only preserve the minima (weak equivalence). If however, we would like to be able to combine strict evidence to our transformed knowledge, we need to perform "preference preserving" transformations. We need "magnitude preserving" transformations (strong equivalence) if we want to combine *any* evidence or give probabilistic interpretation to our transformed knowledge. Most of our transformations in the reminder of this paper are "magnitude preserving" (strongly equivalent). Strong equivalence of two forms of knowledge representation means that the ranking functions that are induced by either these representations are the same (up to a constant difference).
If a PLOFF ψ is strongly equivalent to a network represented by an energy function E then:

1. The set of global minima of E is equal exactly to the set of the preferred models of φ.

2. Both knowledge representations induce the same order on the possible models; i.e., s is "better" than s' iff $Erank_E(s) < Erank_E(s')$ iff $Vrank_\psi(s) < Vrank_\psi(s')$.

3. Knowledge update is cumulative. An addition (update) to the knowledge base can be done by merging the new PLOFF with the existing one. An equivalent result can be obtained by adding the energy terms of the new PLOFF to the energy function representing the old one. The update of a network with a new piece of knowledge is therefore modular and simple.

We define now an equivalence between two calculi.

DEFINITION **4.5** A calculus $\mathcal{C}_1 =< \mathcal{L}, \{0,1\}^n, m >$ is (s-/p-/w-) equivalent to a calculus $\mathcal{C}' =< \mathcal{L}', \{0,1\}^n, m' >$ iff for every $s \in \mathcal{L}$ there exists a (s-/p-/w-) equivalent $s' \in \mathcal{L}'$ and for every $s' \in \mathcal{L}'$ there exists a (s-/p-/w-) equivalent $s \in \mathcal{L}$.

We thus can use the language $\mathcal{L}$ to represent every ranking function that is representable using the language $\mathcal{L}'$, and vice versa. In the sections to come I shall present several equivalent calculi and show that all of them describe the knowledge embedded in SCNs.

4.3. Some examples of equivalent calculi

EXAMPLE **4.3** The calculus of energy functions.

The algebraic notation that was used to describe energy functions as sum-of-products can be viewed as a language for describing ranking functions. The *calculus of energy functions* is therefore $< \{E\}, \{0,1\}^n, m() >$, where $\{E\}$ is the set of all strings representing energy functions written as sum-of-products, and $m(E) = Erank_E$. Two special cases are of particular interest: the calculus of quadratic functions and the calculus of high-order energy functions with no hidden variables.

In section 3.2, algorithms were given that 1) Convert high-order energy functions to strongly equivalent[9] low-order ones with additional hidden variables, and 2) Convert energy functions with hidden variables into strongly equivalent (possibly) higher order ones without those hidden variables. We may therefore conclude that the calculus of high-order energy functions with no hidden units is strongly equivalent to the calculus of quadratic functions. Thus, we can use the language of high-order energy functions with no hidden units to describe any symmetric neural network (SCN) with arbitrary number of hidden units and vice-versa. Note also that the calculus of SCNs whose language describes graphs, weights and thresholds is of course also strongly equivalent to the calculus of quadratic energy functions.

EXAMPLE **4.4** Propositional calculus

In [Pinkas 90a] and in [Pinkas 90b], I showed that the satisfiability of propositional calculus is weakly

[9] In these papers we were concerned only with weak equivalence, but it is easily shown that strong equivalence holds.

equivalent to quadratic energy minimization. The energy function E_{φ} is obtained from φ using the following algorithm:

1. convert the WFF into a conjunction of sub-formulas, each of at most three variables.[10] This is done by adding additional hidden atomic propositions, and "naming" binary subexpressions of the formula using the new propositions. For example, $(((A \vee B) \vee \neg C) \rightarrow (D \vee E))$ is converted into $(T_1 \leftrightarrow A \vee B) \wedge (T_2 \leftrightarrow T_1 \vee \neg C) \wedge (T_2 \rightarrow D \vee E)$.
2. Assuming the result is of the form $\bigwedge_j \varphi_{i_j}$, the energy function is computed to be $\sum_j H_{\neg\varphi_{i_j}}$, where H_{φ} is the characteristic function defined in example 4.1.
3. convert the cubic terms in the result to quadratic ones using a high-order to low-order procedure of section 3.2.[11]

Propositional calculus is therefore weakly equivalent to the calculus of quadratic energy functions and can be used as a high-level language to describe SCNs. However, two limitations exist: 1) The algorithm (in [Pinkas 90b]) that converts an energy function to a satisfiable WFF may generate an exponentially long WFF; and 2) The equivalence is *weak*. It means that although the WFF and the energy function have the same set of satisfying models, neither evidence can be added nor the probabilistic interpretation is preserved.

4.4. The equivalence of penalty logic and SCNs

This section shows that penalty logic and SCNs are strongly equivalent: Every penalty logic formula can be represented *efficiently* in a SCN and every SCN can be described *efficiently* by a penalty logic formula.

4.4.1. Representing penalty logic using SCNs.

Theorem 3 *For every PLOFF $\psi = \{< \rho_i, \varphi_i > \mid i = 1 \dots n\}$ there exists a strongly equivalent quadratic energy function $E(\vec{x}, \vec{t})$; i.e., there exist a constant c such that $Vrank_{\psi} = Erank_E + c$.*

[10] In contrast to the familiar 3-SAT, connectives in a sub-formula are not limited to disjunctions of literals.
[11] This step is not necessary if we allow for high-order (cubic) connectionist networks [Sejnowski 86].

Proof is by construction:

We can construct E from ψ using the following procedure:

1. Start with an empty set of assumptions ψ'. For every pair $< \rho_i, \varphi >$ in ψ, create a new hidden variable T_i, "name" φ_i using $T_i \leftrightarrow \varphi_i$ and add the pairs $< \infty, T_i \leftrightarrow \varphi_i >$ and $< \rho_i, T_i >$ into ψ'. The penalty ∞ represents a real value that is large enough to force the "naming" constraint to be satisfied. The original penalty ρ_i causes the T_i's to compete with each other; while the high penalty ∞ guarantees that if T_i holds (among the winners) then φ_i also holds. ψ' is therefore strongly equivalent to ψ and the T_i's may be considered as hidden variables.

2. Construct the energy function $\sum_i \infty E_{T_i \leftrightarrow \varphi_i} - \sum_j \rho_j T_j$, where E_φ is the function generated by the algorithm described in example 4.4.

The size of the network that is generated is of the same order as the length of the original PLOFF.

The "naming" of the first step is needed only if the number of variables in an assumption φ_i is greater than three. If this is the case and we do not "name" φ_i, then the second step of the algorithm might generate more then one "triple". Each triple will have a penalty that will contribute to the energy function independently of the other triples, and the constraint as a whole will not have the atomicity we expect. Thus, the ranking function that will be generated will not be the one we wished. The high penalty we use for the "naming" causes the system to *always* find solutions that satisfy the "naming" constraints. Once we guarantee that all the "naming" constraints are satisfied, all we need to do is to make the T_i's compete as if they were the original assumptions. When the number of variables is less or equal to three, the way we construct the energy function guarantees that only one triple is generated. Thus, either the constraint is satisfied as a whole (with zero penalty) or it is not satisfied (and the penalty is ρ_i); i.e., the splitting of one constraint into more then one "triple" does not happen, and the atomicity is preserved.

The network that is generated can be seen as performing a search for a preferred model of ψ. According to the sound and complete proof-theory, it can also be seen as searching for a preferred theory of ψ; i.e., the T_i's that win the competition correspond to the assumptions in some preferred theory.

In the following example the assumptions have no more than three variables, thus "naming" is not needed.

Example 4.5 The Nixon diamond case of example 2.1.

The PLOFF that is to be converted is:

$$\psi = \{< 3000, N >, < 1000, N \rightarrow Q >, < 1000, N \rightarrow R >, < 10, Q \rightarrow P >, < 10, R \rightarrow \neg P >\}$$

No "naming" is needed, so $\psi' = \psi$.

Each of the pairs is converted to an energy function:

1000 $N \rightarrow R$	$1000(E_{\neg N \vee R}) = 1000(N - NR)$
1000 $N \rightarrow Q$	$1000(E_{\neg N \vee Q}) = 1000(N - NQ)$
10 $R \rightarrow \neg P$	$10(E_{\neg R \vee \neg P}) = 10(RP)$
10 $Q \rightarrow P$	$10(E_{\neg Q \vee P}) = 10(Q - QP)$
3000 N	$3000(E_N) = 3000(-N)$

Summing the energy terms together:

$$E = -1000NQ - 1000NR + 10RP - 10QP - 1000N + 10Q$$

The corresponding network appears in figure 4.

Example 4.6 Converting the "meeting" example; we first show the general case with "naming" (it is used for demonstration purposes only, since the assumptions have less than four variables):

Penalty	WFF	$E_{\varphi_i}(\vec{x})$
1000	$T_1 \leftrightarrow$ meeting	$1000(T_1 - 2T_1M + M)$
1000	$T_2 \leftrightarrow$ (sick $\rightarrow$($\neg$ meeting))	$1000(T_2SM - 2T_2 - S - M + T_2S + T_2M)$
1000	$T_3 \leftrightarrow$ (cold-only $\rightarrow$ meeting)	$1000(-T_3 - C + 2T_3C + M - T_3M - T_3CM)$
1000	$T_4 \leftrightarrow$ (cold-only $\rightarrow$ sick)	$1000(-T_4 - C + 2T_4C + S - T_4S - T_4CS)$
1	T_1	$-1T_1$
10	T_2	$-10T_2$
100	T_3	$-100T_3$
1000	T_4	$-1000T_4$

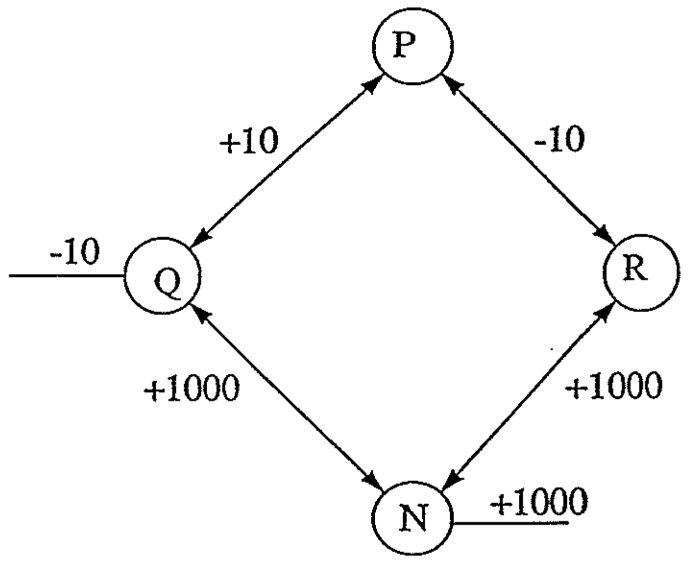

Figure 4: The network that represents the Nixon diamond example. It corresponds to the energy function: $E = -1000NQ - 1000NR + 10RP - 10QP - 1000N + 10Q$

The energy function we get by summing the energy of the assumptions is:

$1000T_2SM - 1000T_3CM - 1000T_4CS - 2000T_1M + 1000T_2S + 1000T_2M + 2000T_3C - 1000T_3M + 2000T_4C - 1000T_4S + 1000M - 2000C + 999T_1 - 2010T_2 - 1100T_3 - 2000T_4$. It is shown as a cubic symmetric network in fig 5-a and as a quadratic network in fig 5-b. Since the assumptions in our example have less than three variables each, we can generate a simpler (strongly equivalent) network from the energy function of $\sum_i \rho_i E_{\neg\varphi_i} = 1(-M) + 100(C - CM) + 1000(C - CS)$ (see fig 5-c).

Once we can generate a network that searches for preferred models (or preferred theories), it is possible to construct a network that will reason according to our definition of entailment. A construction of such network is described in section 6.

4.4.2. representing SCNs as penalty logic formulas. This subsection shows that it is possible to describe efficiently any network by a penalty logic formula. The motivation here is to demonstrate that penalty logic is an efficient and compact language for specification of symmetric connectionist networks.

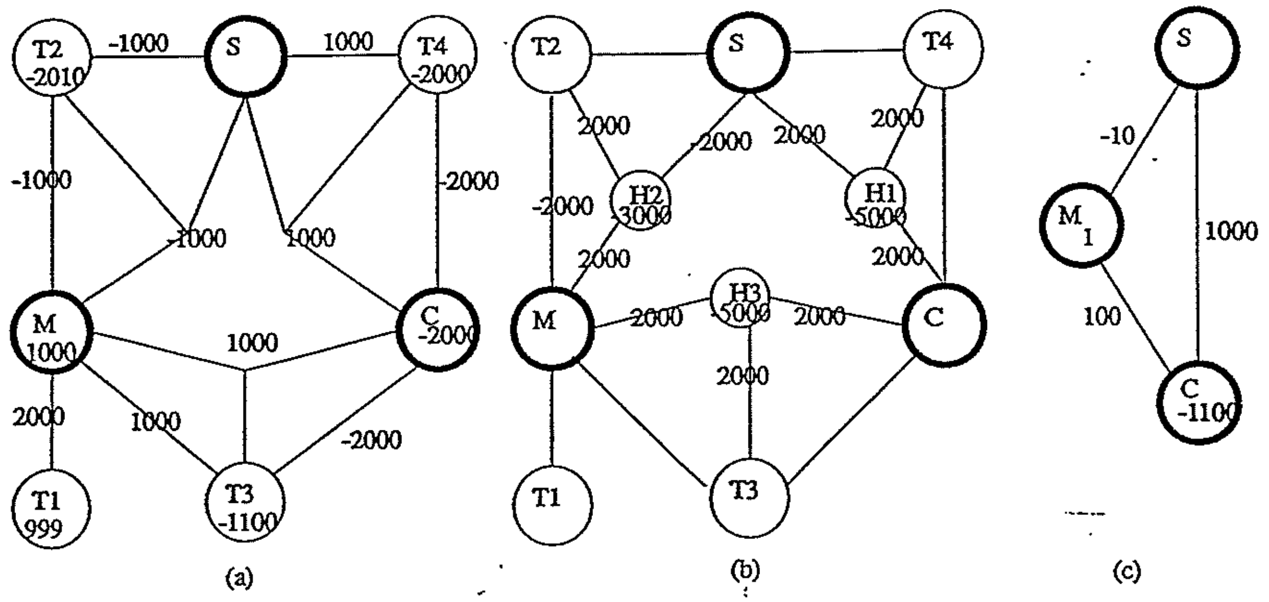

Figure 5: Equivalent symmetric networks for the meeting example (the numbers in the circles are thresholds): a) cubic; b) quadratic; and c) quadratic for the simple conversion (no naming).

Theorem 4 *Every energy function E is strongly equivalent to some PLOFF ψ; i.e., there exists a constant c such that $Erank_E = Vrank_\psi + c$.*

Proof is by construction:

The following algorithm generates a strongly equivalent PLOFF from an energy function:

1. Eliminate hidden variables (if any) from the energy function, using the algorithm of section 3.2.

2. The energy function (with no hidden variables) is now brought into a sum-of-products form and is converted into a PLOFF in the following way:

   Let $E(\vec{x}) = \sum_{i=1}^{m} w_i \prod_{n=1}^{k_i} x_{i_n}$ be the energy function.

   We construct a PLOFF $\psi = \{< -w_i, \bigwedge_{n=1}^{k_i} x_{i_n} > \mid w_i < 0\} \cup \{< w_l, \neg \bigwedge_{n=1}^{k_l} x_{l_n} > \mid w_l > 0\}$.

The formula that is generated is strongly equivalent to the original energy function (network). The size of the formula is in the order of the size of the original network (linear in the number of connections).

EXAMPLE 4.7 Looking at the network of figure 4, we would like to describe this network as a PLOFF: The energy function is:

$$E = -1000NQ - 1000NR + 10RP - 10QP - 1000N + 10Q$$

The negative terms are:

$$< 1000, N \wedge Q >, < 1000, N \wedge R >, < 10, Q \wedge P >, < 1000, N >$$

The positive terms are:

$$< 10, \neg R \vee \neg P >, < 10, \neg Q >$$

The final PLOFF is therefore:

$$< 1000, N \wedge Q >, < 1000, N \wedge R >, < 10, Q \wedge P >, < 1000, N >< 10, \neg R \vee \neg P >, < 10, \neg Q >$$

Note that as it is usually the case with reverse-compilation, the formula we get is not very meaningful; however, it is clear that a compact description exists for every network.

5. Learning propositional formulas

So far, we have learned that networks can be compiled from logic formulas. However, much of the appealing of connectionist models is their ability to learn from examples. This section shows that SCNs can learn unknown propositional formulas inductively, and develop a representation that is equal to the ones constructed by compiling formulas.

Assume the network tries to learn an unknown formula φ by looking at the set of the satisfying models of φ. For simplicity, let us assume that the formula to learn is a satisfiable strict WFF. The task of the network is to update its weights in such a way that at the end of the learning process the energy function is equal to the one obtained by translating φ into E_{φ}. Clearly, by doing so, the set of global minima of the energy function is equal to the set of satisfying models of φ (Γ_{φ}) which is the training set.

Another way to look at the process is as learning of an associative memory: Given a set of vectors to be stored as memories, assuming that those vectors are the satisfying models of some unknown formula, we would like to construct a network such that the global minima of its energy function are exactly equal to the vectors presented.

The algorithm that is described uses hyper-arcs; however, the reader should remember that it is always possible to convert the hyper arcs into pairwise connections by adding hidden units.

DEFINITION **5.1** A k-CNF is a WFF that is formed as a conjunction (AND) of clauses. Each clause is a disjunction (OR) of up to k literals. A literal is either an atomic propositions or a negated ($\neg$) atomic proposition. For example $(A \vee \neg B) \wedge (\neg A \vee \neg C \vee D)$ is a 3-CNF that is composed of two clauses: the first contains two literals and the second contains three.

Note that every formula can be brought into a k-CNF form; thus, the algorithm works in theory for every set of presentations and any unknown formula φ (the algorithm is not practical though, when k is too large).

We present now a new learning rule for symmetric connections (possibly high-order arcs) and a fast learning algorithm that learns an unknown k-CNF formula from the truth assignments that satisfy the formula. After each presentation the network is updated, and the corresponding energy function is guaranteed to have a set of global minima that is exactly equal to the set of presentations seen so far. Therefore, assuming we know the k of the unknown k-CNF formula, the desired network is generated after one cycle over the training set.

Let an instantiation of the visible units $X_1, ..., X_n$ be a vector of zeros and ones $\vec{x} = (x_1, ...x_n)$ such that $x_i \in \{0, 1\}$. A presentation (an example) is an instantiation of the visible units, done by clamping them with the values of the example.

The following learning rule is responsible for the update of the weight of a single l-order (hyper-) arc:

The k-clause learning rule:

Let $Arc = \{X_1, ..., X_l\}$ be a l-order arc.

Given a presentation $\vec{x} = (x_1, ..., x_n)$ that instantiates the visible units to 0/1 values, the l-order arc Arc is updated iff there exists an extension $Ext = (X_{i_1}, ...X_{i_l}, X_{j_1}, ..., X_{j_{k-l}})$, that includes Arc, such that the rest of the units ($Ext - Arc = X_{j_1}, ..X_{j_{k-l}}$) are all instantiated to zeros, and such that the

instantiation $(x_{i_1}, ..., x_{i_l}, x_{j_1}, ..., x_{j_{k-l}})$ is *new*; i.e., never seen in one of the earlier presentations.
If this condition holds, then the weight of arc *Arc* is incremented (+1) if the number of zero units in *Arc* is even (including the all ones case) and is decremented (−1) if the number of zeros is odd.

EXAMPLE **5.1** Given the presentation $ABC = 011$, a 2-clause rule updates the weight of arc AB by $\triangle_{AB} = -1$, since the extension $(A = 0, B = 1)$ is new and the arc contains an odd number of zeros. The weight of BC is updated by $\triangle_{BC} = +1$, since the extension $(B = 1, C = 1)$ is new and the arc contains no zeros. The bias of unit B (which is the singleton arc $\{B\}$) is updated by $\triangle_B = +1$, since the extension $A = 0, B = 1$ is new to the arc B, it is extended using only zeros $(A = 0)$ and the arc B includes no zeros. The weight of A is not updated since it cannot be extended into a set of two units by adding a zero unit. In a similar way, the arc AC is decremented (like AB), and the bias C is incremented (like the bias B).

Note that in a special case of k-clause rule, when $k = 2$ (for pairwise "standard" connections) the rule is very similar to the Hebbian learning rule used so frequently in connectionist learning: When two units have the same value (either both ones or both zeros) the weight is incremented. When the two units have different values (odd number of zeros) the weight of the connection between these units decreases.

Algorithm to learn a k-CNF formula:

Initialize all weights to zero.

For all presentations in the training set

  using the $k-$clause rule, **update** the weights of all the l-order arcs $(0 < l \leq k)$ **until** no more arcs are updated.

Theorem 5 *If the presentations are truth assignments that satisfy some unknown k-CNF formula φ, then the algorithm generates a network whose global minima are* exactly *the set of presentations of the k-CNF formula, and whose energy function is equal E_φ. The network is generated after a* single *pass over the presentations.*

Proof
Only an intuitive proof is given:
The proof is based on showing that the algorithm is equivalent to Valiant's algorithm for learning k-CNF

[Valiant 84]. Valiant's algorithm starts with a list of clauses that consists of all possible k-clauses over n atomic propositions. For every presentation of a truth assignment that satisfies the unknown k-CNF (positive example) the algorithm eliminates those clauses from the list that are not satisfied by the example.

It can be proved that in every step, the conjunction of the clauses in the list is a formula that is consistent with all the presentations seen so far and exactly these presentations. Therefore, when all the examples are seen, the k-CNF formula has been learned.[12]

Show that the set of weight updates performed by the k-clause rule, can be interpreted as clause eliminations. First, show that the conjunction of all possible k-clauses can be represented by a network with zero weights (the initialization step of our algorithm). Later, show that the clause elimination step is equivalent to the set of weight updates performed after each presentation.

An energy function of zero (or any constant energy function) represents formulas that are symmetric conjunction of clause (contradictions or tautologies that have the property that every model satisfies exactly the same number of clauses). The conjunction of all possible clauses in Valiant's algorithm is exactly such formula and therefore, the starting point of our algorithm (with the zero weights) represents the desired list.

The next step is to show that the clause elimination step may be translated to the set of weight updates performed by the k-clause rule.

A clause is eliminated in Valiant's algorithm if it is a disjunction

$$c = \bigvee_{x_i=1} \neg X_i \bigvee_{y_j=0} Y_j$$

where the X_i's are instantiated by the example to one, and the Y_j's are instantiated to zero. A clause elimination is reflected in the energy space by deleting the energy terms of E_c. Since the weights of a network are the energy terms with reverse signs, the weights are actually updated by adding:

$$E_c = H_{\neg c} = H_{\bigwedge X_i \bigwedge \neg Y_j} = \prod_{x_i=1} X_i \prod_{y_j=0} (1 - Y_j).$$

The arcs that are updated as a result of eliminating c contain all the units X_i's in c that are instantiated with ones, while the rest of units are zeros. If the number of zeros in the arc is odd the weight is decremented, and if it is even (or no zeros) it is incremented. Our algorithm updates all the arcs that can be extended into a set of k units by adding zero units. This is equivalent to finding one clause that needs to be deleted and its deletion affects the arc that is updated. The extension $Ext = \{X_1, ..., X_l, X_{j_1}, ..., X_{j_{k-l}}\}$ represents a clause to be eliminated, and when the elimination occurs the $Arc = \{X_1, ..., X_l\}$ is updated according to the parity of the zeros in it. $\square$

EXAMPLE 5.2 Learning the XOR formula $(A \oplus B) \leftrightarrow C$, looking at the four satisfying assignments for $ABC \in \{011, 101, 000, 110\}$.

We need a 3-clause rule since we cannot express our function in less then 3-CNF.

Given the presentation $ABC = 011 \Rightarrow \Delta_{ABC} = -1$ (odd no. of zeros); $\Delta_{BC} = +1$ (even no. of zeros);

given $ABC = 101 \Rightarrow \Delta_{ABC} = -1$ (odd); $\Delta_{AC} = +1$ (even) ;

given $ABC = 000 \Rightarrow \Delta_{ABC} = -1; \Delta_{AC} = +1; \Delta_{BC} = +1; \Delta_{AB} = +1; \Delta_A = -1; \Delta_B = -1; \Delta_C = -1;$

given $ABC = 110 \Rightarrow \Delta_{ABC} = -1; \Delta_{AB} = +1;$

[12] Valiant actually shows that in the PAC model, not all the examples have to be presented.

The energy function obtained by summing the updates (after reversing signs) is $E = 4ABC - 2AC - 2BC - 2AB + A + B + C$. Its global minima are exactly the four presentations, and the high order connection ($4ABC$) can be replaced by second order connections ($4AB - 8AH - 8BH + 8CH + 12H$) by adding one hidden unit H.

An extension of the algorithm using Valiant's notion of "probably approximately correct" (PAC) learning, enables the learning of k-CNF formulas when the k of the formula is not known. The algorithm starts with low k (for example $k = 2$) activates the k-clause learning rule for an entire cycle and then activates a test to see whether the network performs well. If the network passes the test, the algorithm stops; otherwise, k is increased and another cycle begins. The test is a sequence of checks of the network behavior; each tests whether the network converges to a wrong global minimum. The test procedure allows up to m errors within r checks (m and r are computed from the bounds on the error rate requested by the user). If the number of errors is less than m then the test succeeds, otherwise, it fails.

The algorithm guarantees that with arbitrary high probability the network's error rate is arbitrary small. Polynomial learning time and polynomial network complexity are guaranteed if the target concept (the boolean formula) can be represented by a k-CNF with some *constant* k.[13]

The PAC framework allows us to cope with a noisy training set, and to make claims about the generalization capability of the network when not all the presentations are available. In addition, an extension of the algorithm enables the network to learn nonmonotonic knowledge. thus achieving the full representation powers of penalty logic. A full discussion of these algorithms is too lengthly and deserves a separate report.

6. A connectionist inference engine

Suppose a background PLOFF ψ, an evidence PLOFF e, and a query which is a (strict) standard logic WFF φ. We would like to construct a connectionist network to answer one of the possible three answers: 1) $\psi \cup e \models \varphi$; 2) $\psi \cup e \models (\neg\varphi)$; or 3) both $\psi \not\models \varphi$ and $\psi \not\models (\neg\varphi)$ ("ambiguous").

[13]When k is large the size of the network and the performance of the algorithms become exponential in k. Note however, that as is usually the case with knowledge bases, there are many rules and propositions, but each rule is composed of only few number of propositions (a constant k).

Intuitively, our connectionist engine is built from two sub-networks, each of which is trying to find a satisfying model for $\psi\overset{*}{\cup}e$. The first sub-network is biased to search for a preferred model which satisfies also φ, whereas the second sub-network is biased to search for a preferred model which satisfies $\neg\varphi$. If two such models exist, then we conclude that φ is "ambiguous" ($\psi\overset{*}{\cup}e$ entails neither φ nor $\neg\varphi$). If no preferred model also satisfies φ, we conclude that $\psi \cup e \models \neg\varphi$, and if no model also satisfies $\neg\varphi$, we conclude that $\psi \cup e \models \varphi$. For simplicity let us first assume that the evidence e is a strict conjunction of literals (atomic propositions or their negation) and that φ is a single atomic proposition. Later we'll describe a general solution.

To implement this intuition we first need to duplicate our background knowledge ψ and create its copy ψ' by naming all the atomic propositions A using A'. For each atomic proposition Q that might participate in a query, we then add two more propositions: "$QUERY_Q$" and "$AMBIGUOUS_Q$". $QUERY_Q$ is used to initiate a query Q; it will be externally clamped by the user, when he or she inquires about Q. The unit "$AMBIGUOUS_Q$" represents the answer of the system. It will be set to TRUE if we can conclude neither that ψ entails Q nor that ψ entails $\neg Q$.

Our inference engine can be therefore described (in the language of penalty logic) by:

ψ	searches for a preferred model of ψ that satisfies also Q
$\cup\psi'$	searches for a preferred model of ψ that satisfies also $\neg Q$
$\cup\{< \epsilon, (QUERY_Q \rightarrow Q) >\}$	bias ψ to search for a model that satisfies Q
$\cup\{< \epsilon, (QUERY_Q \rightarrow (\neg Q')) >\}$	bias ψ' to search for a model that satisfies $(\neg Q')$
$\cup\{< \epsilon, (Q \wedge \neg Q') \rightarrow AMBIGUOUS_Q >\}$	if two satisfying models exist that do not agree on Q, we conclude "$AMBIGUOUS$"
$\cup\{< \epsilon, (Q \leftrightarrow Q') \rightarrow (\neg AMBIGUOUS_Q) >\}$	if despite the bias we are unable to find two such satisfying models we conclude "$AMBIGUOUS_Q$"

Using the algorithm of Theorem 3, we generate the corresponding network. The network that is generated for the Nixon example is shown in figure 6.

To initiate a query about Q the user externally clamps the unit $QUERY_Q$. This causes a small positive bias ϵ to be sent to unit Q and a small negative bias $-\epsilon$ to be sent to Q'. Each of the two sub-networks ψ and ψ', searches for a global minimum (a satisfying model) of the original PLOFF. The bias (ϵ) is small enough so it does not introduce new global minima for each of the subnetworks. It may however, constrain the set of global minima. If a satisfying model that also satisfies the bias exists, then this model is in the new set of global minima of ψ. The new set of global minima is the set of all preferred

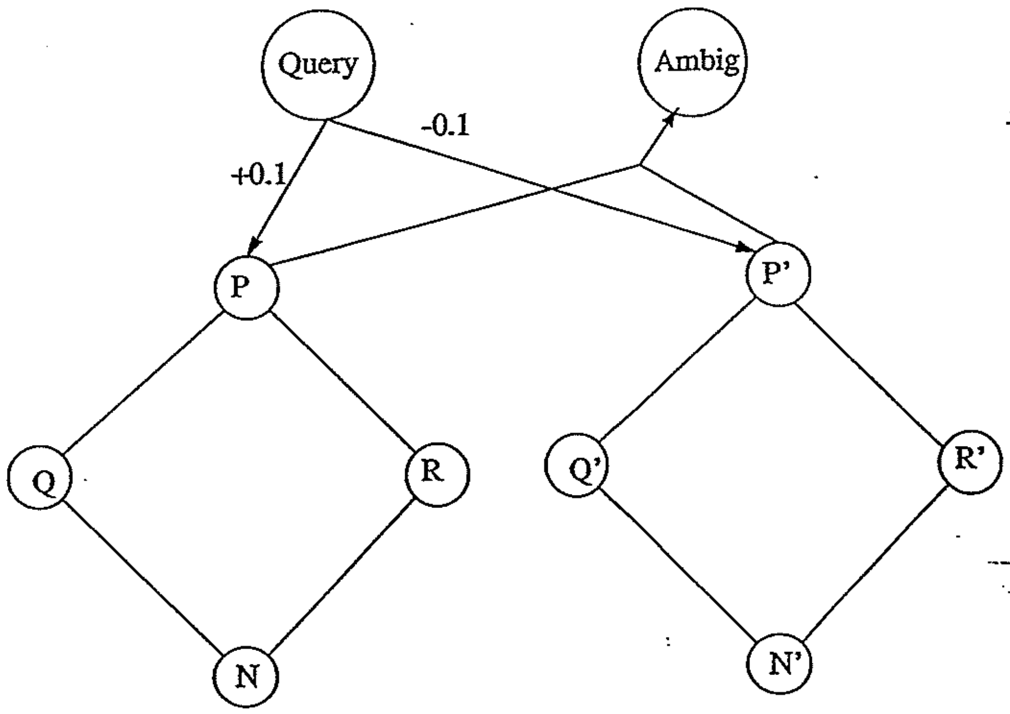

Figure 6: Inference engine for the Nixon diamond case: the two rings represents two similar subnetwork: One searches for a preferred model that satisfies the query and the other searches for a preferred model the falsifies the query.

bias ϵ to be sent to unit Q and a small negative bias $-\epsilon$ to be sent to Q'. Each of the two sub-networks ψ and ψ', searches for a global minimum (a satisfying model) of the original PLOFF. The bias (ϵ) is small enough so it does not introduce new global minima for each of the subnetworks. It may however, constrain the set of global minima. If a satisfying model that also satisfies the bias exists, then this model is in the new set of global minima of ψ. The new set of global minima is the set of all preferred models of ψ that also satisfy the query. If no preferred model also satisfies the query then the set of global minima is unaffected by the bias and the network searches for one of those models (that do not satisfy Q).

The network therefore tries to find models that satisfy also the bias rules. If it succeeds, we conclude *"AMBIGUOUS"*, otherwise we conclude that all the satisfying models agree on the same truth value for the query. The *"AMBIGUOUS"* proposition is then set to "false", and the answer whether $\psi \models \varphi$ or whether $\psi \models \neg\varphi$ can be found in the unit Q. If Q is "true" then the answer is $\psi \models \varphi$ since Q holds in all satisfying models. Similarly, if Q is false, we conclude that $\psi \models \neg\varphi$.

minimum. An annealing schedule[14] like in [Hinton, Sejnowski 86] may be used for such search. A slow enough annealing is certain to find a global minimum and therefore the correct answer, but it might take exponential time. Since the problem is NP-hard, we will probably not find an algorithm that will always give us the correct answer in polynomial time. Traditionally in AI, knowledge representation systems trades the expressiveness of the language they use with the time complexity they allow [Levesque 84],[15] and the accuracy of the answer is usually not sacrificed. The inference mechanism described in this section, as in [Derthick 88], trades the time resources with the accuracy of the answer. Only limited time resources are given, and we wish to stop the search when this limit is reached. The annealing schedule can be planned to fit the time limitation, and an answer is always given at the end of the process. Although the answer may be incorrect, the system is able to improve its guess as more time resources are given.

7. Related work and discussion

7.1. Connectionist approaches

Derthick [Derthick 88] observed that weighted logical constraints (which he called "certainties") can be used in massively parallel architecture. Derthick translated those constraints into special energy functions and used them to implement a subset of the language KL-ONE. The approach described in this paper has a lot of similarities to his system. Looking at his translation from logic to energy functions (Derthick uses different energy functions and no hidden units), there are however, several basic differences: 1) Derthick's "mundane" reasoning is based on finding a most likely single model; his system is never skeptical. The system described in this paper is more cautious and closer in its behavior to symbolic nonmonotonic systems; 2) my system can be implemented with standard low-order units, using relatively well-studied architectures like Hopfield networks or Boltzmann machines. It is possible therefore to take advantage of the hardware implementations as well as of the learning algorithms that were developed for these networks; 3) formal proofs of two-way equivalence are given so that every network can be described as a PLOFF and not just the reverse; 4) a learning algorithm is given that achieves the same networks that are obtained from direct compilation.

[14]There are also other techniques for improving the chances to escape from local minima [Hopfield 84b], [Hinton 89].
[15]Connectionist systems like [Shastri et al. 90] and [Hölldobler 90] trade expressiveness with time complexity.

Another connectionist nonmonotonic system is [Shastri 88]. It uses evidential reasoning based on maximum likelihood to reason in inheritance networks. My approach is different; I use standard low-level connectionist models and am not restricted to inheritance networks.[16] Shastri's system is guaranteed to work and has a polynomial time complexity, whereas the system described here tries to solve an intractable problem and trades correctness with time; i.e., a correct solution (a global minimum) is not guaranteed; however, the chance of finding one improves as more time is given.

This article shares with [Branden 91], [Hölldobler 90], [Shastri et al. 90], and [Touretzky, Hinton 88] the implementationalist motivation [Pinker, Prince 88]. These systems implement subsets of predicate calculus by either spreading activation or by rule-firing. The expressive power of these mechanisms is limited by performance and tractability considerations, and they all stress the problems of representing complex structures, syntax sensitivity and multi-place predicates. In this article I had no intension of attacking these problems; rather, I wanted to show how to represent *any* proprositional constraint, and how networks can cope naturally with inconsistency.

We may look at penalty logic as one of the layers of abstraction that are needed between descriptions of high-level cognitive processes and low-level neural implementations. Thus, penalty logic may be seen as a first level of abstraction that is higher than the neural implementation (see [Branden 91] for a nice discussion on the multi-span approach). Using the language described in this paper we can map[17] several of the systems mentioned above into penalty logic, and then compile them into symmetric networks (possibly by sacrificing efficiency) [Pinkas 91e].

7.2. Symbolic systems

Penalty logic is along the lines of work done in preferential semantics [Shoham 88]. Specifically, systems with preferential semantics that use ranked models, like [Lehmann, Magidor 88], [Lehmann 89] or [Pearl 90]. Lehmann and Magidor's results about the relationship between rational consequence relations and ranked models can be applied to our paradigm: A strict consequence relation (induced

[16] We can easily extend our approach to handle inheritance networks by looking at the atomic propositions as predicates with free variables. Those variables are bound by the user during query time.

[17] Only non oscillating networks may be reduced into SCNs, therefore [Shastri et al. 90] for example cannot be mapped directly.

by a PLOFF ψ) is a binary relation between a strict evidence and a strict conclusion. It is therefore a set of pairs $R_{\psi} = \{<\varphi', \varphi>| \varphi' \overset{\psi}{\models} \varphi\}$, where both φ' and φ are strict WFFs. Lehmann and Magidor defined a *rational* consequence relation as one that satisfies certain conditions (inference rules), and proved that a consequence relation is rational iff it is defined by some ranking function. As a result we may conclude a rather strong conclusion for our system: For every rational consequence relation we can build a ranked model and implement it as a a ranking function on a symmetric network. Also, any symmetric network can be viewed as implementing some rational consequence relation if we use it to determine entailment. We can therefore be sure that every implementation of our inference engine induces a rational consequence relation.

One system of ranked models that can be reduced directly to penalty logic is [Goldszmidt, Pearl 91] which actually computes the penalties from a given conditional knowledge (the user does not specify any penalty) based on maximal entropy considerations. The system uses the same ranking function as the one described in this article.

Penalty logic has some similarities with systems that are based on priorities (given to beliefs). One such system [Brewka 89] is based on levels of reliability. Brewka's system for propositional logic can be mapped into penalty logic by selecting large enough penalties. Systems like [Poole 88] (with strict specificity) can also be implemented using our architecture, and like in [Goldszmidt, et al. 90], the penalties can be generated automatically. Another system that is based on priorities is system Z^+ [Goldszmidt, Pearl 91] where the user does specify the penalties, but there is a "ghost" that changes them so that several nice properties hold (e.g. specificity). Penalty logic can only approximate priority systems by assigning penalties that reflect scaled priorities.[18] Every conclusion that is entailed in a priority system like system Z^+, will also be entailed by the approximating penalty logic knowledge base. However, some conclusions that are ambiguous in a priority system may be drawn decisively in penalty logic. In this sense penalty logic can be considered as bolder (less cautious) than those which are based on priorities.

For example consider the "penguins and the wings" case [Goldszmidt, Pearl 91]. We are given the following defaults: birds fly; birds have wings; penguins are birds and penguins do not fly. Many

[18]The penalties are scaled so that there is no subset of low-priority assumptions whose sum exceeds a higher priority.

systems based on priorities (like Z^+) will not be able to conclude that penguins have wings. Penalty logic in contrast will conclude according to our intuition; i.e., that penguins do have wings despite the fact that penguins do not fly. The reason for this intuitive deduction is that penalty logic considers the models where penguins do not fly but have wings to be more "normal" than models where penguins do not fly and have no wings (like in [Goldszmidt, et al. 90]). Priority-based system will be ambiguous since they don't have such preference.

For another example consider the Nixon case (example 2.1) when we add to it: $< 1000, N \rightarrow FF >$ and $< 10, FF \rightarrow \neg P >$ (Nixon is also a football fan and football fans tend to be not pacifist). Most other nonmonotonic systems will still be skeptical about P [Touretzky 86], [Loui 87], [Geffner 89] [Pearl 90], [Lehmann 89]. Our system boldly, and in contrast with intuition, decides $\neg P$ since it is better to defeat the one assumption supporting P than the two assumptions supporting $\neg P$. We can correct this behavior however, by multiplying the penalty for $Q \rightarrow P$ by two. Further, a network like our system that learns, may adjust the penalties autonomously and thus develop its own intuition and nonmonotonic behavior.

Because we do not allow for arbitrary partial orders ([Shoham 88] [Geffner 89]) of the models, there are other fundamental[19] problematic examples where our system (and all systems with ranked models semantics) boldly concludes, while other systems are skeptical (these are cases where the intuition tell us that skepticism is the right behavior).

The following is an example for which we have clear intuition; nevertheless, no ranking function exists that induces the intuitive behavior we wish:

EXAMPLE **7.1** Assume the following defeasible rules: $A \rightarrow D$, $B \rightarrow \neg D$ and $C \rightarrow \neg D$. The intuition we have states that:
Given A, C, D we should conclude $\neg B$; therefore, $rank(A\bar{B}CD) < rank(ABCD)$.
Given A, B, C we should conclude that D is ambiguous; therefore, $rank(ABCD) = rank(ABC\bar{D})$.
Given $A, C, \bar{D}$ we should conclude that B is ambiguous; therefore, $rank(ABC\bar{D}) = rank(A\bar{B}C\bar{D})$.
Given $A, \bar{B}, C$ we should conclude that D is ambiguous; therefore, $rank(A\bar{B}C\bar{D}) = rank(A\bar{B}CD)$.
This is a contradiction since $rank(A\bar{B}CD) < .rank(A\bar{B}CD)$.

[19]This claim is due to Hector Geffner (private communication).

8. Conclusions

The main task of this paper was to develop the theoretical foundations needed for a connectionist inference engine that is capable of representing and learning nonmonotonic knowledge. Along these lines I have introduced penalty logic and showed mappings between its sentences and SCNs.

Penalty logic may be used as a framework for defeasible reasoning and handling inconsistency. Several systems can be mapped to this paradigm and therefore suggest settings of the penalties. When the right penalties are given (for example using algorithms like in [Brewka 89] that are based on specificity), penalty calculus features a non-monotonic behavior that (usually) matches our intuition. It is possible to show, though, that some intuitions cannot be expressed in ranking-functions.

A strong equivalence between sentences of the logic and symmetric networks is formally proved. This two-way equivalence serves two purposes: 1) we can translate a sentence of penalty logic into an equivalent network; this serves the basic construction of our inference engine; 2) any symmetric network (and also asymmetric non oscillating networks) can be described by penalty logic sentences. The calculus may thus be used as a specification language and gives another clarifying look at the dynamics of such networks.

Several equivalent high-level languages can be used to describe SCNs: 1) quadratic energy functions; 2) high-order energy functions with no hidden units; 3) propositional logic, and finally 4) penalty logic. All these languages are expressive enough to describe any SCN and every sentence of such languages can be translated into a SCN; however, penalty logic has properties that make it more attractive than the other languages. Algorithms are given for translating between any two of the knowledge representation forms above.

An inference engine constructed that is capable of answering whether a query follows the knowledge or not. When a query is clamped, the global minima of such network correspond exactly to the correct answer. Using massively parallel hardware convergence should be very fast, although the worst case for the *correct* answer is still exponential. The mechanism however trades the probability of getting a sound answer with the time given to solve the problem.

The engine can obtain its knowledge either by compiling a symbolic knowledge base or by learning propositional rules inductively by looking at examples. Learning is shown to be equivalent to a powerful symbolic algorithm developed within the PAC paradigm.

Revision of the knowledge base and adding new evidence are easy tasks if we use penalty logic to describe the network: adding (or deleting) a PLOFF is simply computing the energy function of the new PLOFF and then adding (deleting) it to the background energy function. A local change to the PLOFF describing the network is translated to a local change in the network.

The mappings given in this paper are limited to propositional knowledge; however, their potential exceeds the propositional case, and allows also for higher level paradigms (like first-order logic) to be represented [Pinkas 91e].

Acknowledgment: Thanks to Bill Ball, Jon Doyle, Hector Geffner, Sally Goldman, Dan Kimura, Stan Kwasny, Fritz Lehmann, Ron Loui, Judea Pearl and Dave Touretzky for helpful discussions.

References

[Branden 91] J.A. Branden, "Encoding complex symbolic data structures with some unusual connectionist techniques," in J.A Branden and J.B. Pollack, *Advances in Connectionist and Neural Computation Theory 1,* High-level connectionist models, Ablex Publishing Corporation, 1991.

[Brewka 89] G. Brewka, "Preferred sub-theories: An extended logical framework for default reasoning," *Proceedings of IJCAI,* pp. 1043-1048, 1989.

[Derthick 88] M. Derthick "Mundane reasoning by parallel constraint satisfaction," PhD thesis, CMU-CS-88-182 Carnegie Mellon University, Sept. 1988

[Feldman 85] J.A Feldman "Energy and the behavior of connectionist models," technical report, Computer Science Department, University of Rochester, TR-155 1985.

[Fodor, Pylyshyn 88] J.A. Fodor, Z.W. Pylyshyn, "Connectionism and cognitive architecture: A critical analysis," *Cognition 28,* pp. 3-71, 1988.

[Geffner 89] H. Geffner, "Defeasible reasoning: Causal and conditional theories," PhD Thesis, Department of Computer Science, UCLA, 1989.

[Goldszmidt, et al. 90] M. Goldszmidt, P. Morris, J. Pearl, "A maximum entropy approach to non-monotonic reasoning," *Proceedings of AAAI,* pp 646-652, 1990.

[Goldszmidt, Pearl 91] M. Goldszmidt, J. Pearl, "System Z^+: A formalism for reasoning with variable-strength defaults," *Proceedings of AAAI,* pp 399-404, 1991.

[Hinton 89] G.E Hinton "Deterministic Boltzmann learning performs steepest descent in weight space," *Neural Computation 1,* no. 1, 1989.

[Hinton, Sejnowski 86] G.E Hinton and T.J. Sejnowski, "Learning and re-learning in Boltzman Machines," in J. L. McClelland and D. E. Rumelhart, *Parallel Distributed Processing: Explorations in The Microstructure of Cognition I,* pp. 282 - 317, MIT Press, 1986.

[Hinton 90] G.E. Hinton, "Preface to the special issue on connectionist symbol processing," *Artificial Intelligence 46*, nos. 1-2, 1990.

[Hölldobler 90] S. Hölldobler, "CHCL, a connectionist inference system for Horn logic based on connection method and using limited resources," International Computer Science Institute TR-90-042, 1990.

[Hopfield 82a] J. J. Hopfield "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences 79,* pp. 2554-2558, 1982.

[Hopfield 84b] J. J. Hopfield "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences 81,* pp. 3088-3092, 1984.

[Hopfield, Tank 85] J.J. Hopfield, D.W. Tank "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics 52*, pp. 144-152.

[Lang 89] T.E. Lang and M.G. Dyer, "High-level inferencing connectionist network," *Connection Science 1,* no. 2, pp. 181-217, 1989

[Lehmann, Magidor 88] D. Lehmann, M. Magidor, "Rational logics and their models: A study in cumulative logic," technical report TR-86-16, Leibnitz Center for Computer Science, Hebrew University, Jerusalem, 1988.

[Lehmann 89] D. Lehmann, "What does a conditional knowledge base entail?," *Proc. of the International Conf. on Knowledge Representation and reasoning,* pp. 212-222, Toronto, Canada, 1989.

[Levesque 84] H.J. Levesque, "A fundamental tradeoff in knowledge representation and reasoning," *Proc. CSCSI-84*, pp. 141-152, London, Ontario, 1984.

[Lifschitz 85] V. Lifschitz, Computing circumscription. *Proceedings of IJCAI*, 1985.

[Loui 87] R.P. Loui, "Defeat among arguments: A system of defeasible inference," *Computational Intelligence 3,* no. 3, 1987.

[McCarthy 80] J. McCarthy, "Circumscription, a form of nonmonotonic reasoning," *Artificial Intelligence 25*, pp. 41-72, 1980.

[Pearl 90] J. Pearl, "System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning," in proceedings *TARK-90*, M. Vardi (ed.), pp. 121-135, 1990.

[Pinkas 90a] G. Pinkas, "Energy minimization and the satisfiability of propositional calculus," technical report, Department of Computer Science, Washington University, WUCS-90-03, 1990.

[Pinkas 90b] G. Pinkas, "Energy minimization and the satisfiability of propositional calculus," *Neural Computation 3,* no. 2, 1991.
Also in Touretzky, D.S., Elman, J.L. Sejnowski, T.J. Hinton, G.E. (eds), *Proceedings of the 1990 Connectionist Models Summer School*, San Mateo, Morgan Kaufmann.

[Pinkas 91c] G. Pinkas, "Propositional Non-Monotonic Reasoning and Inconsistency in Symmetric Neural Networks," *Proceedings of IJCAI*, Sydney, 1991.

[Pinkas 91d] G. Pinkas, "Representing first-order- predicate logic in symmetric networks," to appear in *Advances in Neural Information Processing Systems IV* (NIPS), 1992.

[Pinkas 91e] G. Pinkas, "Converting binary threshold networks into symmetric networks," technical report, Computer Science Department, Washington University, WUCS-91-31, 1991.

[Pinkas 91f] G. Pinkas, "Symbolic knowledge representation using symmetric connectionist networks," PhD thesis, Washington University, in preparation, 1992.

[Pinker, Prince 88] S. Pinker, A. Prince, "On language and connectionism: Analysis of a parallel distributed processing model of language acquisition," *Cognition 28*, pp. 73-193, 1988.

[Poole 85] D. Poole, "On the comparison of theories: preferring the most specific explanation," *Proceedings of IJCAI*, pp. 144-147, 1985.

[Poole 88] D. Poole, "A logical framework for default reasoning," *Artificial Intelligence 36*, 1988.

[Rescher, Manor 70] N. Rescher,R. Manor, "On inference from inconsistent premises," *Theory and Decision 1*, pp. 179-217, 1970.

[Reiter 80] R. Reiter, "A logic for default reasoning," *Artificial Intelligence 13*, pp. 81-132, 1980.

[Rumelhart et al. 86] D.E. Rumelhart, G.E Hinton, J.L. Mcclelland, "A general framework for parallel distributed processing," in J. L. McClelland and D. E. Rumelhart, *Parallel Distributed Processing: Explorations in The Microstructure of Cognition I*, MIT Press, 1986.

[Sejnowski 86] T. J. Sejnowski "Higher-Order Boltzman Machines," Neural Networks for Computing, *Proceedings of The American Institute of Physics 151*, Snowbird Utah, pp. 3984, 1986.

[Shastri 88] L. Shastri, "Semantic networks: An evidential formulation and its connectionist realization," Pitman, London, 1988.

[Shastri et al. 90] L. Shastri, V. Ajjanagadde, "From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings," technical report, University of Pennsylvania, Philadelphia, MS-CIS-90-05, 1990.

[Shoham 88] Y. Shoham, *Reasoning about Change*, MIT Press, Cambridge, 1988.

[Shortliffe 76] E.H Shortliffe, *Computer-based medical consultation, MYCIN*, Elsevier, New York, 1976.

[Simari, Loui 90] G. Simari, R.P. Loui , "Mathematics of defeasible reasoning and its implementation," to appear in *AI Journal.*

[Smolensky 86] P. Smolensky, "Information processing in dynamic systems: Foundations of harmony theory," in J.L.McClelland and D.E.Rumelhart, *Parallel Distributed Processing: Explorations in The Microstructure of Cognition I* , MIT Press, 1986.

[Touretzky 86] D.S. Touretzky, *The Mathematics of Inheritance Systems*, Pitman, 1986.

[Touretzky, Hinton 88] D.S Touretzky, G.E. Hinton "A distributed connectionist production system," *Cognitive Science 12,* 3, pp.423-466, 1988.

[Valiant 84] L.G. Valiant, "A theory of the learnable," *Communications of the ACM 27*, pp. 1134-1142, 1984.