CMU-CS-86-138

A Calculus for Inheritance in Monotonic Semantic Nets

Richmond H. Thomason
John F. Horty
David S. Touretzky

July 1986

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

CMU-CS-86-138

A Calculus for Inheritance in Monotonic Semantic Nets

Richmond H. Thomason

Linguistics Department
University of Pittsburgh
Pittsburgh, PA 15260

John F. Horty

Philosophy Department
Carnegie-Mellon University
Pittsburgh, PA 15213

David S. Touretzky

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

This material is based on work supported by the National Science Foundation under Grant No. IST-8516313.

CONTENTS

1. Introduction

This report is devoted to a theoretical analysis of inheritance in monotonic semantic nets with positive and negative links, but without relations. To a logician, the context will seem unusually simple (there are, for instance, no genuine propositional connectives); but the texture of the resulting logic is surprisingly rich, given the impoverishment of the background language.

We establish several basic results. After setting out the fundamental ideas behind monotonic inheritance in Sections 2 and 3, we characterize the inheritance relation, in Section 4, through a sequent calculus, or natural-deduction system, that is relatively natural and well-motivated.[1] It is often thought that the logic of monotonic semantic nets, at least, is simply the classical predicate calculus. We show that this is not so. The logic we present is non-classical; and in Section 5, it is proved to be both sound and complete with respect to monotonic inheritance. In Section 6, turning from proof-theory to semantics, we show that the inheritance relation is equivalent also to a notion of validity arising from interpretations over a certain four-valued matrix that has been thought before to have some computational significance.

One traditional attraction of inheritance networks has always been their natural correspondence with graphs, which makes them particularly appropriate as vehicles of knowledge representation for concurrent computing architectures, where graph-searches can be very fast. In Section 7, we present inference algorithms over monotonic semantic nets for such a concurrent architecture, the Parallel Marker Propagation Machine defined by Scott Fahlman. The algorithms are proved to be correct and complete.

2. Notation

Letters from the beginning of the alphabet (a, b, c) will represent *objects*, and letters from the middle of the alphabet (p, q, r) will represent *kinds* of objects. Letters from the end of the alphabet (u, v, w, x, y, z) will range over both objects and kinds.

An *assertion* will have the form $x \rightarrow\!\!\!\!\!\cdot\; y$ or $x \not\rightarrow y$, where y is a kind. If x is an object,

[1] The logic we describe here is reminiscent in many ways of *syllogistic*, the dominant framework for formal logic for over a millenium. See [9] for a historical overview; for more recent work on syllogistic, see, for example, [14] or [4].

these assertions should be interpreted as ordinary atomic statements: $a \rightarrow p$ and $b \not\rightarrow p$, for example, might represent the statements 'Jumbo is an elephant' and 'Tweety isn't an elephant'. If x is a kind, the assertions should be interpreted as necessarily true generic statements: $p \rightarrow q$ and $r \not\rightarrow q$ might represent the statements 'Elephants are mammals' and 'Birds aren't mammals'. Of course, in general, generic statements can be true even in the presence of exceptions. 'Birds fly' is a true generic, even though penguins don't; 'Mammals don't lay eggs' is true even though the platypus does. In this paper, however, we will limit our attention to the special class of generic statements which, if true, are true without exception, by necessity or definition.

Capital Greek letters (Γ, Δ, Π) will represents *nets*, where a net consists of a set I of individuals and a set K of kinds, together with a set of positive links and a set of negative links (both subsets of $(I \times K) \cup (K \times K)$). We can identify the positive and negative links in a net with our positive and negative assertions. This analogy between assertions and links in a net will be exploited throughout this report. It enters into our notation as well. Since there is no difference in informal interpretation between a link $a \rightarrow p$ and a singular formula pa, we will use these notations interchangeably. Similarly for $a \not\rightarrow p$ and $\neg pa$.

A link is said to be *monotonic* if it represents the kind of generic statement that can only be true if it is true without exception. In this report, then, we assume that all links are monotonic. Likewise, we consider only *monotonic nets*, those containing none but monotonic links.

3. Monotonic inheritance

The relation of *inheritance* holds between a net Γ and the individual links *supported* by Γ. Roughly, the idea is that an individual link is supported by a net Γ if it must be true whenever all the links in Γ are true, or if it is true in the situation represented by Γ. Since we are focusing on monotonic nets, it is reasonable to impose some conditions on inheritance. In fact, let's *define* it. Where '$\models$' represents the inheritance relation, the set of links supported by a monotonic net Γ will be the smallest superset of Γ closed under the following

conditions:

$$(3.1)\ \Gamma \models p \rightarrow p;$$

$$(3.2)\ \Gamma \models x \rightarrow q,\ \Gamma \models q \rightarrow r \ \Rightarrow\ \Gamma \models x \rightarrow r;$$

$$(3.3)\ \Gamma \models p \not\rightarrow q \ \Rightarrow\ \Gamma \models q \not\rightarrow p;$$

$$(3.4)\ \Gamma \models x \rightarrow q,\ \Gamma \models q \not\rightarrow r \Rightarrow \Gamma \models x \not\rightarrow r;$$

$$(3.5)\ \Gamma \models p \rightarrow q,\ \Gamma \models a \not\rightarrow q \Rightarrow \Gamma \models a \not\rightarrow p.$$

The following facts about monotonic inheritance will be useful for establishing results later in this report, relating inheritance to more familiar logical ideas. They may also have some independent interest.

Definition 1: A *monotonic path* is a pair of sequences $\langle x_0, \ldots, x_n\ ;\ y_1, \ldots, y_m\rangle$, where the first sequence $\langle x_0, \ldots, x_n\rangle$ is nonempty and the second sequence $\langle y_1, \ldots, y_m\rangle$ may be empty. A monotonic path is *permitted* by a net Γ if (1) for all i, $0 \leq i < n$, $x_i \rightarrow x_{i+1} \in \Gamma$ or x_i is a kind and $x_i = x_{i+1}$, (2) either $x_n \not\rightarrow y_1 \in \Gamma$ or $y_1 \not\rightarrow x_n \in \Gamma$, and (3) for all i, $1 \leq i < m$, $y_{i+1} \rightarrow y_i \in \Gamma$ or y_i is a kind and $y_i = y_{i+1}$. A path is *positive* if its second sequence is empty, and is *negative* otherwise. A positive path $\langle x_0, \ldots, x_n\rangle$ *enables* the positive link $x_0 \rightarrow x_n$, and a negative path $\langle x_0, \ldots, x_n\ ;\ y_1, \ldots, y_m\rangle$ *enables* the negative link $x_0 \not\rightarrow y_m$.

Lemma 1: $\Gamma \models A$ iff Γ permits a monotonic path that enables A.

Proof. It is easy to show by induction on the length of a monotonic path that if Γ permits a path that enables A then $\Gamma \models A$. On the other hand, it is also straightforward to check that the set of links enabled by monotonic paths permitted by Γ is closed under Conditions (3.1) to (3.5). For example, Condition (3.3) follows from the fact that if Γ permits the path $\langle x_0, \ldots, x_n\ ;\ y_1, \ldots, y_m\rangle$, and x_0 is a kind, then Γ also permits the path $\langle y_m, \ldots, y_1\ ;\ x_n, \ldots, x_0\rangle$. Likewise, Condition (3.4) follows from the fact that: if Γ permits both the paths $\langle u_0, \ldots, u_i\rangle$ and $\langle x_0, \ldots, x_n; y_1, \ldots, y_m\rangle$, then if $u_i = x_0$, Γ also permits the path $\langle u_0, \ldots, u_{i-1}, x_0, \ldots, x_n; y_1, \ldots, y_m\rangle$. ∎

Most of the things we want to prove about the monotonic inheritance relation follow from this fundamental Lemma, which allows us to reason about the links that a net supports by reasoning about the paths it permits.

Lemma 2: If $\Gamma \models A$ then $\Gamma \cup \Delta \models A$. (In other words, monotonic nets have the monotonicity property.)

Proof. By induction on the length of monotonic paths permitted by Γ. ∎

Lemma 3: If $\Gamma \models A$, then $\Gamma \cup \{A\} \models B$ iff $\Gamma \models B$.

Proof. It follows from the previous Lemma that $\Gamma \cup \{A\} \models B$ if $\Gamma \models B$. So suppose that $\Gamma \models A$, and that $\Gamma \cup \{A\} \models B$. There are four cases to consider: (i) both A and B are positive assertions, (ii) A is positive and B is negative, (iii) B is positive while A is negative, and (iv) both A and B are negative. The idea in each case is that, if A occurs as a basic link in the path permitted by $\Gamma \cup \{A\}$ enabling B, it can be replaced by the path enabling A itself, which contains only links from Γ. We illustrate only the cases (ii) and (iii).

In case (ii), Lemma 1 tells us that Γ permits a positive path $\langle x_0, \ldots, x_n \rangle$ enabling A, and that $\Gamma \cup \{A\}$ permits a negative path $\langle u_0, \ldots, u_i; v_1, \ldots v_j \rangle$ enabling B. If A occurs as a basic link in the path enabling B, then either (ii$a$) $x_0 = u_k$ and $x_n = u_{k+1}$, for some k, $0 \leq k < n$, or (ii$b$) $x_0 = v_{k+1}$ and $x_n = v_k$, for some k, $1 \leq k < j$. If (ii$a$) then we can see from Definition 1 that Γ itself will permit the path $\langle u_0, \ldots, u_{k-1}, x_0, \ldots, x_n, u_{k+2}, \ldots, u_i; v_1, \ldots, v_j \rangle$, which also enables B, so that $\Gamma \models B$, by Lemma 1. Likewise, if (ii$b$), then Γ permits the path $\langle u_0, \ldots, u_i; v_1, \ldots v_{k-1}, x_n, \ldots, x_0, v_{k+2}, \ldots, v_j \rangle$, which enables B, so that $\Gamma \models B$. Of course, if A does not occur as a basic link in the path enabling B (i.e., if neither (ii$a$) nor (ii$b$)), then Γ already permits the path $\langle u_0, \ldots, u_i; v_1, \ldots v_j \rangle$ enabling B.

In case (iii), the link A cannot occur in the path permitted by $\Gamma \cup \{A\}$ that enables B, since a negative link cannot occur in a positive path; so the same path enabling B must be permitted by Γ itself, and we are done. ∎

Lemma 4: Where Δ consists only of links having the form $a \rightarrow p$ or $a \not\rightarrow p$, $\Gamma \models p \rightarrow q$ if $\Gamma \cup \Delta \models p \rightarrow q$, and $\Gamma \models p \not\rightarrow q$ if $\Gamma \cup \Delta \models p \not\rightarrow q$. (In other words, adding new "particular" links to a net will not lead it to support new "generic" links.)

Proof. By induction on the length of monotonic paths permitted by $\Gamma \cup \Delta$. ∎

Lemma 5: Let Γ^a be a net not containing the individual a. Then $\Gamma^a \models p \rightarrow q$ if $\Gamma^a \cup \{a \rightarrow p\} \models a \rightarrow q$, and $\Gamma^a \models p \not\rightarrow q$ if $\Gamma^a \cup \{a \rightarrow p\} \models a \not\rightarrow q$.

Proof. Suppose $\Gamma^a \cup \{a \rightarrow p\} \models a \rightarrow q$. By Lemma 1, $\Gamma^a \cup \{a \rightarrow p\}$ permits a path

$\langle x_0, x_1, \ldots, x_n \rangle$, with $x_0 = a$ and $x_n = q$. By Definition 1, we know, since x_0 is not a kind, that $x_0 \rightarrow x_1 \in \Gamma^a \cup \{a \rightarrow p\}$. Hence, by the restriction on Γ^a, $x_1 = p$. As we can see also from Definition 1, since $\Gamma^a \cup \{a \rightarrow p\}$ permits the path $\langle x_0, x_1, \ldots, x_n \rangle$, it must permit the subpath $\langle x_1, \ldots, x_n \rangle$ enabling the link $p \rightarrow q$. By Lemma 1, then, $\Gamma^a \cup \{a \rightarrow p\} \models p \rightarrow q$; so by Lemma 4, since $a \rightarrow p$ is a "particular" link and $p \rightarrow q$ is "generic", we have $\Gamma^a \models p \rightarrow q$. The argument that $\Gamma^a \models p \not\rightarrow q$ if $\Gamma^a \cup \{a \rightarrow p\} \models a \not\rightarrow q$ is similar. ∎

4. A sequent calculus

We formulate a calculus for proving sequents $\Gamma \vdash A$, where Γ is a set of formulas and A a formula. Informally, this means that Γ has enough information to yield A, or something like that.

Since we are considering only monotonic semantic nets, all of whose links represent statements true without exception, it might seem natural to suppose that that we could embed these nets in the classical predicate calculus by translating:

$a \rightarrow p$ as pa,

$a \not\rightarrow p$ as $\neg pa$,

$p \rightarrow q$ as $\forall x[px \rightarrow qx]$, and

$p \not\rightarrow q$ as $\forall x[px \rightarrow \neg qx]$.

However, this is wrong. Even though we are considering only monotonic links, we are still forced to move to a nonclassical logic in order to capture the net interpretation of formulas. For instance, we wouldn't want to be able to prove the classically valid sequent $a \rightarrow p, a \not\rightarrow p \vdash a \rightarrow q$, since the net $\Gamma = \{a \rightarrow p, a \not\rightarrow p, a \not\rightarrow q\}$, shows us that $\{a \rightarrow p, a \not\rightarrow p\} \not\models a \rightarrow q$.

Examples such as this seem to have been generally overlooked. In fact, it seems to be a kind of "folk theorem" in artificial intelligence that the logic of semantic networks, and even frame systems, is just the classical first-order predicate calculus; often, Pat Hayes's [7] is cited in support of this claim. Though it is true that some simple theories formulable in first-order logic—for instance, $\{Fa, \forall x[Fx \rightarrow Gx], \forall x[Fx \rightarrow \neg Hx]\}$—can be coded using semantic nets, it is hard to think of a natural general relation according to which the two would be equivalent. Quantifiers and classical disjunction can't be expressed

easily in nets, and of course there is no straightforward way to reproduce nonmonotonic reasoning in classical first-order logic. What this example shows, though, is that even without disjunction, quantifiers, or conjunction, and even though inheritance is construed as monotonic, there is a clear sense in which logical consequence over nets diverges from the consequence relation of classical logic.

Our sequent calculus contains two *structural rules*, as follows.

$$\Gamma \vdash a \to p \text{ if } a \to p \in \Gamma$$

$$\Gamma \vdash a \not\to p \text{ if } a \not\to p \in \Gamma$$

These give us the axioms. In addition, we have *logical rules*, for introducing both $\to$ and $\not\to$, on the right and on the left of the turnstile.

$$\frac{\Gamma^a, a \to p \vdash a \to q}{\Gamma^a \vdash p \to q} \vdash \to$$

$$\frac{\Gamma \vdash a \to p \quad \Gamma, a \to q \vdash A}{\Gamma, p \to q \vdash A} \to \vdash \qquad \frac{\Gamma \vdash a \not\to q \quad \Gamma, a \not\to p \vdash A}{\Gamma, p \to q \vdash A} \to \vdash'$$

$$\frac{\Gamma^a, a \to p \vdash a \not\to q}{\Gamma^a \vdash p \not\to q} \vdash \not\to$$

$$\frac{\Gamma \vdash a \to p \quad \Gamma, a \not\to q \vdash A}{\Gamma, p \not\to q \vdash A} \not\to \vdash \qquad \frac{\Gamma \vdash a \to q \quad \Gamma, a \not\to p \vdash A}{\Gamma, p \not\to q \vdash A} \not\to \vdash'$$

In the rules $\vdash \to$ and $\vdash \not\to$, Γ^a is supposed to represent a collection of formulas not containing a. We do need both the rules $\to \vdash$ and $\to \vdash'$ to capture the meaning of $\to$ on the left of the turnstile; neither will do alone. Likewise, both $\not\to \vdash$ and $\not\to \vdash'$ are necessary.

To illustrate these rules, we provide here a sample proof, of the sequent $p \rightarrow q, q \not\rightarrow r \vdash p \not\rightarrow r$.

$$\dfrac{\dfrac{\dfrac{a \rightarrow p \vdash a \rightarrow p \quad a \rightarrow q \vdash a \rightarrow q}{a \rightarrow p, p \rightarrow q \vdash a \rightarrow q} \rightarrow \vdash \qquad a \not\rightarrow r \vdash a \not\rightarrow r}{p \rightarrow q, q \not\rightarrow r, a \rightarrow p \vdash a \not\rightarrow r} \not\rightarrow \vdash}{p \rightarrow q, q \not\rightarrow r \vdash p \not\rightarrow r} \vdash \not\rightarrow$$

Lemma 6: The following rule of weakening is admissible:

$$\dfrac{\Gamma \vdash A}{\Gamma, \Delta \vdash A}$$

Proof. By induction on length of proof in the sequent calculus. ∎

5. Soundness and completeness

Semantic nets play a dual conceptual role: we can think of them either as *models* or as *theories* (i.e., structured sets of assumptions, together with procedures for inferring conclusions from these assumptions). This may seem surprising to those who are used to the crisp division in traditional logic between semantic and proof-theoretic ideas. Semantic nets, however, come from a different tradition, in which the distinction is not so clear: it is harder to draw the line, in knowledge representation, between what is represented and what is doing the representing.

The analogy between nets and theories should be obvious: the links contained in a net are like hypotheses, or axioms; the links supported by a net are like the consequences derivable from a set of hypotheses; and the paths permitted by a net are like proofs. (The analogy between monotonic paths and proofs, in particular, will be exploited in the Lemmas leading up to the proof of Theorem 2.) To understand their role as models, it might be helpful to place semantic nets against the background of the epistemic tradition

in interpreting logics.[2] This tradition is largely Bayesian, and *quantitative*: the core of the enterprise consists in interpreting formalized languages by assigning probabilities to formulas. (Probabilities also lead a double life, since they can also be thought of as objective chances; but here we are thinking of the subjective interpretation of probability.) Though, mathematically, semantic nets are very different from probability functions, they are like these functions at least in providing something that can be claimed to depict the structure of an agent's knowledge. As models, then, semantic nets can be regarded as supplying a *qualitative* epistemic interpretation of a logical calculus, in which valid inferences can be characterized in terms of what is known in arbitrary epistemic states.[3]

Thinking of nets, now, as interpretations, or models, we show in this Section that the sequent calculus defined in Section 4 is both sound and complete with respect to the inheritance relation defined in Section 3.

Soundness first.

Theorem 1: If $\Gamma \vdash A$ is provable, then $\Gamma \models A$.

Proof. By induction on length of proof of $\Gamma \vdash A$. Obviously, it is trivial if $\Gamma \vdash A$ is an axiom. The inductive cases are all straightforward; we will illustrate only a couple.

Suppose the conclusion $\Gamma \vdash p \not\rightarrow q$ comes by an application of the rule $\vdash \not\rightarrow$, with premise $\Gamma^a, a \rightarrow p \vdash a \not\rightarrow q$. By inductive hypothesis, we know that $\Gamma^a \cup \{a \rightarrow p\} \models a \not\rightarrow q$. Since a is a fresh individual, however, Lemma 5 tells us that $\Gamma \models p \not\rightarrow q$.

Suppose the conclusion $\Gamma \vdash p \rightarrow q$ comes by an application of the rule $\rightarrow \vdash$ from the premises $\Gamma \vdash a \rightarrow p$ and $\Gamma, a \rightarrow q \vdash A$. By inductive hypothesis, we know that $\Gamma \models a \rightarrow p$ and that $\Gamma \cup \{a \rightarrow q\} \models A$. Lemma 1 then tells us that Γ permits a positive path $\langle u_0, \ldots u_n \rangle$, with $u_0 = a$ and $u_n = p$; and also that $\Gamma \cup \{a \rightarrow q\}$ permits some path enabling A, positive or negative depending on the character of A, which we will write

---

[2] See [16], [10], and [5].

[3] One difficulty with the probabilistic approach to epistemic semantics is that probability functions are so close to Boolean assignments that it is hard to see how to arrange things so that the epistemic style of semantics is interestingly different from the classical one. Semantic nets do not have this problem. This is especially true of nets with nonmonotonic inheritance, but—as we have seen—there are nontrivial differences even in the monotonic case. Also, semantic nets begin to provide ways of thinking about matters that are not available in familiar types of models, including probabilistic ones. For instance, a net provides a setting in which a conclusion can have a limited number of possible reasons.

as $\langle v_0, \ldots v_n[; w_1, \ldots w_m]\rangle$, with the square-bracketed part denoting an optional negative segment. Now there are two cases to consider: either (i), $v_0 = a$ and $v_1 = q$ (i.e., the path enabling A begins with the link $a \rightarrow q$), or (ii) $v_0 \neq a$ or $v_1 \neq q$. In case (ii), Γ also supports the path $\langle v_0, \ldots v_n[; w_1, \ldots w_m]\rangle$; so $\Gamma \models A$ by Lemma 1; so $\Gamma \cup \{p \rightarrow q\} \models A$, by Lemma 2, and we are done. In case (i), we replace the direct link between a and q with the permitted path from a to p and the direct link between p and q, to get the path $\langle u_0, \ldots u_n, v_1, \ldots v_n[; w_1, \ldots w_m]\rangle$. Inspection of Definition 1 tells us that $\Gamma \cup \{p \rightarrow q\}$ permits this new path, which also enables A; so $\Gamma \cup \{p \rightarrow q\} \models A$ by Lemma 1, and we are done. ∎

To prove that the sequent calculus is complete with respect to the inheritance relation, we need to show that $\Gamma \vdash A$ is provable whenever $\Gamma \models A$. The idea behind our proof is simple. If $\Gamma \models A$, we know that Γ permits some monotonic path that enables A. What we provide here, roughly, is a recipe, defined by induction on the length of such a path, for transforming it into a sequent calculus proof of $\Gamma \vdash^{\cdot} A$.

Definition 2: If σ is a positive path $\langle x_0, \ldots x_n\rangle$, the sequence of nets $\Pi(\sigma)_1, \ldots, \Pi(\sigma)_n$ corresponding to the initial subpaths of σ is given as follows:

$$\Pi(\sigma)_1 = \{x_0 \rightarrow x_1\},$$

$$\Pi(\sigma)_{i+1} = \begin{cases} \Pi(\sigma)_i, & \text{if } x_{i+1} = x_i; \\ \Pi(\sigma)_i \cup \{x_i \rightarrow x_{i+1}\}, & \text{otherwise.} \end{cases}$$

The net $\Pi(\sigma)$ is then defined to be $\Pi(\sigma)_n$.

It is easy to see that this definition gives us the *minimum* net permitting a positive path.

Lemma 7: If σ is a positive path, $\Pi(\sigma)$ permits σ; and for any net Γ permitting σ, $\Pi(\sigma) \subseteq \Gamma$.

Proof. By Definition 1 and induction on the length of σ. ∎

Things are more complicated in the case of negative paths, since there is no minimum net permitting such a path, but rather, two *minimal* nets.

Definition 3: Let τ be the negative path $\langle x_0, \ldots, x_n; y_1, \ldots, y_m\rangle$, containing the path σ as its positive part. We define the two sequences of nets $\Pi'(\tau)_1, \ldots, \Pi'(\tau)_m$ and $\Pi''(\tau)_1, \ldots, \Pi''(\tau)_m$, as follows. First, let

$$\Pi'(\tau)_1 = \Pi(\sigma) \cup \{x_n \not\rightarrow y_1\},$$

$$\Pi''(\tau)_1 = \Pi(\sigma) \cup \{y_1 \not\rightarrow x_n\}.$$

Then, where $\Pi^*(\tau)_i$ is either $\Pi'(\tau)_i$ or $\Pi''(\tau)_i$, let

$$\Pi^*(\tau)_{i+1} = \begin{cases} \Pi^*(\tau)_i, & \text{if } y_{i+1} = y_i; \\ \Pi^*(\tau)_i \cup \{y_{i+1} \rightarrow y_i\}, & \text{otherwise.} \end{cases}$$

The net $\Pi'(\tau)$ is defined to be $\Pi'(\tau)_m$, and $\Pi''(\tau)$ is defined to be $\Pi''(\tau)_m$.

Neither $\Pi'(\tau)$ nor $\Pi''(\tau)$ can include the other, since one contains the link $x_n \not\rightarrow y_1$ while the other contains $y_1 \not\rightarrow x_n$. But they are otherwise alike, and they are the minimal nets permitting τ.

Lemma 8: If τ is a negative path, then both $\Pi'(\tau)$ and $\Pi''(\tau)$ permit τ; and for any Γ permitting τ, either $\Gamma \subseteq \Pi'(\tau)$ or $\Gamma \subseteq \Pi''(\tau)$.

Proof. Again, by Definition 1 and induction on the length of τ. ∎

Lemma 9: Let σ be the positive path $\langle x_0, \ldots, x_n \rangle$, with x_0 an individual. Then $\Pi(\sigma) \vdash x_0 \rightarrow x_n$ is provable.

Proof. We show by induction that for each i, $1 \leq i \leq n$, there exists a proof $\mathcal{P}_i$ of the sequent $\Pi(\sigma)_i \vdash x_0 \rightarrow x_i$.

First, take $i = 1$. Then $\Pi(\sigma)_i \vdash x_0 \rightarrow x_i$ is $x_0 \rightarrow x_1 \vdash x_0 \rightarrow x_1$, an axiom. Let this axiom be $\mathcal{P}_1$.

Next, supposing $\mathcal{P}_i$ proves $\Pi(\sigma)_i \vdash x_0 \rightarrow x_i$, we construct a proof $\mathcal{P}_{i+1}$ of the sequent $\Pi(\sigma)_{i+1} \vdash x_0 \rightarrow x_{i+1}$. There are two cases to consider: (i) $x_{i+1} = x_i$, (ii) $x_{i+1} \neq x_i$. In case (i), $\Pi(\sigma)_{i+1}$ is identical with $\Pi(\sigma)_i$, and the statement $x_0 \rightarrow x_{i+1}$ is identical with the statement $x_0 \rightarrow x_i$; so we need only let $\mathcal{P}_{i+i}$ be $\mathcal{P}_i$, and we are done. In case (ii), let $\mathcal{P}_{i+1}$ be

$$\dfrac{\dfrac{\mathcal{P}_i}{\Pi(\sigma)_i \vdash x_0 \rightarrow x_i} \qquad \Pi(\sigma)_i, x_0 \rightarrow x_{i+1} \vdash x_0 \rightarrow x_{i+1}}{\Pi(\sigma)_i, x_i \rightarrow x_{i+1} \vdash x_0 \rightarrow x_{i+1}} \rightarrow \vdash$$

This gives us what we want, since the right premise is an axiom, and the conclusion is the sequent $\Pi(\sigma)_{i+1} \vdash x_0 \rightarrow x_{i+1}$. ∎

Lemma 10: Let τ be the negative path $\langle x_0, \ldots, x_n; y_1, \ldots y_m \rangle$, with x_0 an individual. Then both $\Pi'(\tau) \vdash x_0 \not\rightarrow y_m$ and $\Pi''(\tau) \vdash x_0 \not\rightarrow y_m$ are provable.

Proof. Where $\sigma = \langle x_0, \ldots, x_n \rangle$ is the positive part of τ, the previous Lemma gives us a proof $\mathcal{P}$ of the sequent $\Pi(\sigma) \vdash x_0 \to x_n$. We show by induction that for each i, $1 \leq i \leq n$, $\mathcal{P}$ can be extended to a proof $\mathcal{P}_i'$ of the sequent $\Pi'(\tau)_i \vdash x_0 \not\to y_i$, and also to a proof $\mathcal{P}_i''$ of the sequent $\Pi''(\tau)_i \vdash x_0 \not\to y_i$.

Take $i = 1$. Then $\Pi'(\tau)_i \vdash x_0 \not\to y_i$ is $\Pi(\sigma), x_n \not\to y_1 \vdash x_0 \not\to y_1$; so we let $\mathcal{P}_1'$ be

$$\dfrac{\dfrac{\mathcal{P}}{\Pi(\sigma) \vdash x_0 \to x_n} \qquad \Pi(\sigma), x_0 \not\to y_1 \vdash x_0 \not\to y_1}{\Pi(\sigma), x_n \not\to y_1 \vdash x_0 \not\to y_1} \not\to \vdash$$

This gives us a proof, since the right premise is an axiom. Likewise, when $i = 1$, $\Pi''(\tau)_i \vdash x_0 \not\to y_i$ will be $\Pi(\sigma), y_1 \not\to x_n \vdash x_0 \not\to y_1$; so we can let $\mathcal{P}_1''$ be

$$\dfrac{\dfrac{\mathcal{P}}{\Pi(\sigma) \vdash x_0 \to x_n} \qquad \Pi(\sigma), x_0 \not\to y_1 \vdash x_0 \not\to y_1}{\Pi(\sigma), y_1 \not\to x_n \vdash x_0 \not\to y_1} \not\to \vdash'$$

Again, this gives us the proof we want, since the right premise is an axiom.

The inductive step of the proof can be handled uniformly. As before, we let $\Pi^*(\tau)_i \vdash x_0 \not\to y_i$ be either $\Pi'(\tau)_i \vdash x_0 \not\to y_i$ or $\Pi''(\tau)_i \vdash x_0 \not\to y_i$; and we let $\mathcal{P}_i^*$ be either $\mathcal{P}_i'$ or $\mathcal{P}_i''$. Supposing that $\mathcal{P}_i^*$ proves $\Pi^*(\tau)_i \vdash x_0 \not\to y_i$, then, we show how to construct a proof $\mathcal{P}_{i+1}^*$ of the sequent $\Pi^*(\tau)_{i+1} \vdash x_0 \not\to y_{i+1}$. Just as in the previous Lemma, there are two cases to consider: (i) $y_{i+1} = y_i$, and (ii) $y_{i+1} \neq y_i$. In case (i), $\Pi^*(\tau)_{i+1}$ is identical with $\Pi^*(\tau)_i$, and the statement $x_0 \not\to y_{i+1}$ is identical with the statement $x_0 \not\to y_i$; so we can let $\mathcal{P}_{i+1}^*$ be $\mathcal{P}_i^*$. In case (ii), let $\mathcal{P}_{i+1}^*$ be

$$\dfrac{\dfrac{\mathcal{P}_i^*}{\Pi^*(\tau)_i \vdash x_0 \not\to y_i} \qquad \Pi^*(\tau)_i, x_0 \not\to y_{i+1} \vdash x_0 \not\to y_{i+1}}{\Pi^*(\tau)_i, y_{i+1} \to y_i \vdash x_0 \not\to y_{i+1}} \to \vdash'$$

This gives us the desired proof, since the right premise is an axiom. ∎

Lemma 11: Let σ be the positive path $\langle x_0, \ldots, x_n \rangle$, with x_0 a kind. Then $\Pi(\sigma) \vdash x_0 \to x_n$ is provable.

Proof. Let σ^a be the path $\langle a, x_0, \ldots, x_n \rangle$, with a an individual. Evidently $\Pi(\sigma^a) = \Pi(\sigma) \cup \{a \rightarrow x_0\}$. By Lemma 9, since a is an individual, we have a proof $\mathcal{P}$ of the sequent $\Pi(\sigma^a) \vdash a \rightarrow x_n$; that is, of the sequent $\Pi(\sigma), a \rightarrow x_0 \vdash a \rightarrow x_n$. So we extend this proof as follows

$$\dfrac{\dfrac{\mathcal{P}}{\Pi(\sigma), a \rightarrow x_0 \vdash a \rightarrow x_n}}{\Pi(\sigma) \vdash x_0 \rightarrow x_n} \vdash \rightarrow$$

to get a proof of the desired conclusion. ∎

Lemma 12: Let τ be the negative path $\langle x_0, \ldots, x_n; y_1, \ldots, y_m \rangle$, with x_0 a kind. Then both $\Pi'(\tau) \vdash x_0 \not\rightarrow y_m$ and $\Pi''(\tau) \vdash x_0 \not\rightarrow y_m$ are provable.

Proof. Similar to the proof of the previous Lemma. The proof is uniform, so again, we let $\Pi^*(\tau)$ represent either $\Pi'(\tau)$ or $\Pi''(\tau)$. Let τ^a be the path $\langle a, x_0, \ldots, x_n; y_1, \ldots, y_m \rangle$, with a an individual. Evidently, $\Pi^*(\tau^a) = \Pi^*(\tau) \cup \{a \rightarrow x_0\}$. By Lemma 10, since a is in individual, we have a proof $\mathcal{P}$ of the sequent $\Pi^*(\tau^a) \vdash a \not\rightarrow y_m$; that is, of the sequent $\Pi^*(\tau), a \rightarrow x_0 \vdash a \not\rightarrow y_m$. So we extend this proof as follows

$$\dfrac{\dfrac{\mathcal{P}}{\Pi^*(\tau), a \rightarrow x_0 \vdash a \not\rightarrow y_m}}{\Pi^*(\tau) \vdash x_0 \not\rightarrow y_m} \vdash \not\rightarrow$$

to get a proof of the desired conclusion. ∎

Theorem 2: If $\Gamma \models A$, then $\Gamma \vdash A$ is provable.

Proof. There are four cases: (i) A is a positive statement concerning an individual, of the form $a \rightarrow p$; (ii) A is a negative statement concerning an individual, of the form $a \not\rightarrow p$; (iii) A is a positive statement concerning a kind, of the form $p \rightarrow q$; (iv) A is a negative statement concerning a kind, of the form $p \not\rightarrow q$. We show in each of these four cases that $\Gamma \vdash A$ is provable if $\Gamma \models A$.

Case (i). Since $\Gamma \models a \rightarrow p$, we know by Lemma 1 that Γ permits a path $\sigma = \langle x_0, \ldots, x_n \rangle$, with $x_0 = a$ and $x_n = p$. Lemma 9 tells us that $\Pi(\sigma) \vdash a \rightarrow p$ is provable. But $\Pi(\sigma) \subseteq \Gamma$ by Lemma 7; so by Lemma 6, $\Gamma \vdash a \rightarrow p$ must be provable as well.

Case (ii). Since $\Gamma \models a \not\rightarrow p$, we know by Lemma 1 that Γ permits a path $\tau = \langle x_0, \ldots, x_n; y_1, \ldots, y_m \rangle$, with $x_0 = a$ and $y_m = p$. Lemma 10 tells us that both $\Pi'(\tau) \vdash a \not\rightarrow p$ and $\Pi''(\tau) \vdash a \not\rightarrow p$ are provable. But we must have either $\Pi'(\tau) \subseteq \Gamma$ or $\Pi''(\tau) \subseteq \Gamma$, by Lemma 8; so in either case, Lemma 6 tells us that $\Gamma \vdash a \not\rightarrow p$ must be provable as well.

Cases (iii) and (iv) are similar to cases (i) and (ii), using Lemmas 11 and 12, respectively, instead of Lemma 9 and 10. ∎

6. A four-valued interpretation

The language with which we have been dealing is so weak that it is incapable of representing many of the principles that typically distinguish classical from nonclassical logics. In particular, Excluded Middle can't be expressed, since disjunction is not available. Still the invalidity in nets of inferences like

$$(*) \quad a \rightarrow p, a \not\rightarrow p \vdash a \rightarrow q,$$

which was mentioned in Section 4, where an invalidating net was provided, forced us to resort to a nonclassical proof-theory. In this Section, looking at nets from a different perspective, we supply a nonclassical semantic interpretation (model theory) to accompany our proof-theory for monotonic inheritance.

Examples such as (*) suggest, not only that we will need to look for a nonclassical interpretation of negation, but also that the models that have been developed in connection with *relevance logic*[4] should provide materials for an interpretation. Though the original motivation for relevance logic was not computational, it did grow out of a belief that "fallacies of relevance" such as (*) should not be regarded as logically valid. More recently, several people have suggested that relevance logic may have applications in computer science. Nuel Belnap has argued, in [2] and [3], that a certain four-valued matrix, originally developed to characterize the valid inferences in a fragment of relevance logic, might be useful also as a guide for reasoning about information stored in databases. And a number of computer scientists have explored applications of relevance logic to knowledge representation and nonmonotonic reasoning.[5] As far as we know, however, no relation

[4] See [1].

[5] See [13], [12], and [11].

as direct as the one we will establish below has been established between concepts from relevance logic and structures that arise naturally within artificial intelligence.

Along with Belnap, we take as our truth values the four members of the set $\mathcal{T} = \{\{\mathrm{T}\}, \{\mathrm{F}\}, \emptyset, \{\mathrm{T}, \mathrm{F}\}\}$; and we interpret these truth values likewise, to represent the four information states of a system with respect to a proposition p, considered as data: (i) the state of possessing evidence for p, and no evidence to the contrary; (ii) the state of possessing evidence against p, and no evidence for p; (iii) the state of possessing no evidence either for or against p; (iv) the state of possessing evidence for p, and evidence against p as well. These explanations should suggest why it's useful to take the power set of $\{\mathrm{T}, \mathrm{F}\}$ as our set of truth values: if X is one of the values from $\mathcal{T}$, '$\mathrm{T} \in X$' means that there is evidence for any proposition bearing the truth value X, and '$\mathrm{F} \in X$' means that there is evidence against such a proposition.

We imagine a set D of individuals, the domain. A *valuation* v on the language we have described, relative to a domain D, assigns an individual $v(a)$ in D to each individual term a of the language, and a function $v(p)$ from D to $\mathcal{T}$ to each generic term p. Where v is a valuation, v^{d}/a is the valuation like v for all terms other than a, but which assigns the value d to a. The following rules extend v to the entire language.

(6.1) $v(pa) = [v(p)](v(a))$.

(6.2) $v(\neg pa) = \mathrm{Not}(v(pa))$, where $\mathrm{Not}(\{\mathrm{T}\}) = \{\mathrm{F}\}$, $\mathrm{Not}(\{\mathrm{F}\}) = \{\mathrm{T}\}$, $\mathrm{Not}(\emptyset) = \emptyset$, and $\mathrm{Not}(\{\mathrm{T}, \mathrm{F}\}) = \{\mathrm{T}, \mathrm{F}\}$.

(6.3) $v(p \rightarrow q) = \{\mathrm{T}\}$ if for all $\mathrm{d} \in \mathrm{D}$, we have $\mathrm{T} \in v^{\mathrm{d}}/a(qa)$ if $\mathrm{T} \in v^{\mathrm{d}}/a(pa)$ and $\mathrm{F} \in v^{\mathrm{d}}/a(pa)$ if $\mathrm{F} \in v^{\mathrm{d}}/a(qa)$; and $v(p \rightarrow q) = \emptyset$ otherwise.

(6.4) $v(p \not\rightarrow q) = \{\mathrm{T}\}$ if for all $\mathrm{d} \in \mathrm{D}$, we have $\mathrm{F} \in v^{\mathrm{d}}/a(qa)$ if $\mathrm{T} \in v^{\mathrm{d}}/a(pa)$ and $F \in v^{\mathrm{d}}/a(pa)$ if $\mathrm{T} \in v^{\mathrm{d}}/a(qa)$; and $v(p \not\rightarrow q) = \emptyset$ otherwise.

Note that conditions (6.3) and (6.4) assign only the values $\{\mathrm{T}\}$ and $\emptyset$ to generic statements $p \rightarrow q$ and $p \not\rightarrow q$. Nothing more is required, since the language doesn't provide for negations of these statements.

The separation of the truth conditions of positive statements from those of negative statements makes it possible to characterize a number of operations that could be used to interpret statements of the form $p \rightarrow q$ (and statements of the form $p \not\rightarrow q$ also). This proliferation of alternatives, which sometimes makes it difficult to motivate interpretations,

is one of the penalties of working with a many-valued logic. In this four-valued logic, we can define at least three conditional-like operations on functions P, Q from D to $\mathcal{T}$.

- $f(P,Q) = \{\mathrm{T}\}$ if for all $\mathrm{d} \in \mathrm{D}$, $\mathrm{T} \in Q(\mathrm{d})$ if $\mathrm{T} \in P(\mathrm{d})$; and $f(P,Q) = \emptyset$ otherwise.
- $g(P,Q) = \{\mathrm{T}\}$ if for all $\mathrm{d} \in \mathrm{D}$, $\mathrm{F} \in P(\mathrm{d})$ if $\mathrm{F} \in Q(\mathrm{d})$; and $g(P,Q) = \emptyset$ otherwise.
- $h(P,Q) = \{\mathrm{T}\}$ if for all $\mathrm{d} \in \mathrm{D}$, $\mathrm{T} \in Q(\mathrm{d})$ if $\mathrm{T} \in P(\mathrm{d})$ and $\mathrm{F} \in P(\mathrm{d})$ if $\mathrm{F} \in Q(\mathrm{d})$; and $h(P,Q) = \emptyset$ otherwise.

These operations are not equivalent. For instance, if $\mathrm{D} = \{\mathrm{d}\}$, $P(\mathrm{d}) = \emptyset$, and $Q(\mathrm{d}) = \{F\}$, then $f(P,Q) = \{T\}$, and $g(P,Q) = \emptyset$. We choose to interpret statements like $p \rightarrow q$ through the operation h, which conjoins f and g. This is a strong inferential connection, enabling more conceptual connections than the other two. It seems appropriate, since both "positive" and "contrapositive" inferences are validated in the case of monotonic inheritance. (If ostriches without exception are birds, then we can conclude both that Tweety is a bird if we are told that he is an ostrich, and that he is not an ostrich if we are told that he is not a bird.)

Given the four-valued interpretation, we define semantic implication in the usual way, and also a relation of equivalence between nets and valuations.

Definition 4: Γ *implies* A if for all valuations v, if $\mathrm{T} \in v(B)$ for all $B \in \Gamma$, then $\mathrm{T} \in v(A)$.

Definition 5: A net Γ is *equivalent* to a valuation v if $\Gamma \models A$ iff $\mathrm{T} \in v(A)$.

To show that this implication relation coincides with the relation of monotonic inheritance, we first establish some Lemmas about nets and their equivalent valuations.

Lemma 13: For every valuation v, there is an equivalent net Γ.

Proof. Where v is a valuation on a domain D, let Γ be defined by letting $a \rightarrow p \in \Gamma$ iff $\mathrm{T} \in v(pa)$, $a \not\rightarrow p \in \Gamma$ iff $\mathrm{F} \in v(pa)$, $p \rightarrow q \in \Gamma$ iff $\mathrm{T} \in v(p \rightarrow q)$, and $p \not\rightarrow q \in \Gamma$ iff $\mathrm{T} \in v(p \not\rightarrow q)$.

Since Γ has been defined so that $x \rightarrow y \in \Gamma$ if $\mathrm{T} \in v(x \rightarrow y)$, it follows at once that $\Gamma \models x \rightarrow y$ if $\mathrm{T} \in v(x \rightarrow y)$; and it is easy to see by considering conditions (3.1) to (3.5) that $\mathrm{T} \in v(x \rightarrow y)$ if $\Gamma \models x \rightarrow y$. We will give two of the five cases: (3.3) and (3.4). For (3.3), note the symmetric form of the valuation rule (6.4), for $v(p \not\rightarrow q)$; this ensures that $\mathrm{T} \in v(q \not\rightarrow p)$ if $\mathrm{T} \in v(p \not\rightarrow q)$. For (3.4), suppose that $x = p$, and that we have $\mathrm{T} \in v(p \rightarrow q)$ and $\mathrm{T} \in v(q \not\rightarrow r)$. Now if $\mathrm{T} \in v^{\mathrm{d}}/a(pa)$, then by (6.3) $\mathrm{T} \in v^{\mathrm{d}}/a(qa)$, so by (6.4) $\mathrm{F} \in v^{\mathrm{d}}/a(ra)$. On the other hand, if $\mathrm{T} \in v^{\mathrm{d}}/a(ra)$, then by (6.4) $\mathrm{F} \in v^{\mathrm{d}}/a(qa)$, so by

(6.3) $F \in v^d/a(pa)$. Therefore, by (6.4), we have $T \in v(p \not\rightarrow r)$. The argument is similar in case $x = a$. ∎

Lemma 14: For every net Γ, there is an equivalent valuation v.

Proof. Where Γ is a net, let Γ' be the result of first adding new individual terms a_p to the language of Γ, one new term for each generic term of the original language, and then adding to Γ a link $a_p \rightarrow p$ for each generic term p. Clearly, Γ' is a conservative extension of Γ: if A involves no new terms, then $\Gamma' \models A$ iff $\Gamma \models A$. Moreover, $\Gamma' \models qa_p$ iff $\Gamma \models p \rightarrow q$ and $\Gamma' \models \neg qa_p$ iff $\Gamma \models p \not\rightarrow q$. Thus, a_p is a witness for p in the sense that a_p has a property in the new net if and only if that property belongs through inheritance in the net to any individual that has p. (Such a "generic witness" is something that could not be constructed, of course, in the context of a classical logic.)

Now construct a valuation v on domain I', where I' is the set of individual terms of the expanded language, by letting $v(a) = a, T \in [v(p)](a)$ iff $\Gamma' \models pa$, and $F \in [v(p)](a)$ iff $\Gamma' \models \neg pa$. We show v equivalent to Γ'; it follows that v is equivalent to Γ.

For A having the form pa, we have $T \in v(A)$ iff $\Gamma' \models A$ by definition; for A having the form $\neg pa$, $T \in v(A)$ iff $F \in [v(p)](a)$ iff $\Gamma' \models \neg pa$. In particular, we have $T \in v(qa_p)$ iff $\Gamma' \models qa_p$, and $F \in v(qa_p)$ iff $\Gamma' \models \neg qa_p$. For A having the form $p \rightarrow q$, the satisfaction definition gives us $T \in v(A)$ iff

(I) for all $a \in I'$, $T \in v(qa)$ if $T \in v(pa)$, and $F \in v(pa)$ if $F \in v(qa)$.

Suppose that $T \in v(A)$; then $T \in v(qa_p)$, so $\Gamma' \models qa_p$, so $\Gamma' \models p \rightarrow q$. On the other hand, suppose that $\Gamma' \models p \rightarrow q$; then for all $a \in I'$, if $\Gamma' \models pa$ then $\Gamma' \models qa$, and if $\Gamma' \models \neg qa$ then $\Gamma' \models \neg pa$. So (I) follows. For A having the form $p \not\rightarrow q$, the satisfaction definition gives us $T \in v(A)$ iff

(II) for all $a \in I'$, $F \in v(qa)$ if $T \in v(pa)$, and $F \in v(pa)$ if $T \in v(qa)$.

Suppose $T \in v(A)$. Then $F \in (qa_p)$, so $\Gamma' \models \neg qa_p$, so $\Gamma' \models p \not\rightarrow q$. On the other hand, suppose that $\Gamma' \models p \not\rightarrow q$. Then for all $a \in I'$, if $\Gamma' \models pa$ then $\Gamma' \models \neg qa$, so that if $T \in v(pa)$ then $F \in v(qa)$. But also then $\Gamma' \models q \not\rightarrow p$, and so for all $a \in I'$, if $\Gamma' \models qa$ then $\Gamma' \models \neg pa$, so that if $T \in v(qa)$ then $F \in v(pa)$. (II) follows. ∎

These Lemmas allow us to show that implication in the four-valued logic characterizes the inheritance relation.

Theorem 3: $\Gamma \models A$ iff Γ implies A.

Proof. Suppose $\Gamma \not\models A$. By Lemma 14, there is a valuation v equivalent to the net Γ. For all $B \in \Gamma$, then, $\mathrm{T} \in v(B)$, but $\mathrm{T} \notin v(A)$; so Γ doesn't imply A. On the other hand, suppose Γ doesn't imply A, so that there exists a valuation v such that $\mathrm{T} \in v(B)$, for all $B \in \Gamma$, but $\mathrm{T} \notin v(A)$. By Lemma 13, there is a net Δ equivalent to the valuation v: so for all $B \in \Gamma$, $\Delta \models B$; and $\Delta \not\models A$. It follows by successive applications of Lemma 3 (as many applications as there are members of Γ) that $\Delta \cup \Gamma \not\models A$; and from this it follows by Lemma 2 that $\Gamma \not\models A$. ∎

7. Parallel marker propagation algorithms

Semantic inheritance networks are attractive as formalisms for knowledge representation in part because their natural correspondence with graphs allows certain important kinds of inference to be carried out by simple graph-searching algorithms. One especially elegant approach results when we view inference algorithms as a species of *graph coloring* algorithms. The best known AI system in which inference is performed by graph coloring is Scott Fahlman's NETL [6], which introduced a novel computer architecture known as a Parallel Marker Propagation Machine (PMPM); in this section, we define marker propagation algorithms on this machine for performing inheritance reasoning over monotonic nets.

A PMPM is an automaton composed of active elements that play the roles of nodes and links in a graph. Each element has a small number of internal states (maker bits, which can be on or off, representing the presence or absence of markers), and a limited ability to communicate information to the elements with which it is connected. The nodes and links in a PMPM are responsive to several marker propagation commands, each of which directs the assignment of markers to particular nodes, often by "propagating" them from one node to another through the intervening links. PMPM's are SIMD (Single Instruction stream, Multiple Data stream) machines: marker propagation commands are broadcast globally to all elements and executed in parallel by the elements to which they apply. Parallel marker propagation algorithms can be described in terms of marker propagation commands; the result of executing such an algorithm in a particular net is a *coloring*—a static assignment of colors to nodes—that can be used to convey some information about

the net.

We take as our only two markers the usual truth values, T and F. Now it is standard practice (see [15]) to let the markers themselves serve as colors, so that the marker propagation commands can be seen also as propagating colors directly. In the present context, however, it is more natural to take as colors the four members of the matrix $\mathcal{T} = \{\{T\}, \{F\}, \emptyset, \{T,F\}\}$, explored in Section 6. We define the color assigned to a node by a marker propagation algorithm as the unique member of $\mathcal{T}$ containing just those markers placed by the algorithm on that node; each algorithm will then result in a total coloring of each net, with no more than one color assigned to any particular node.

The notation used here for specifying marker propagation commands is that of [15]. Commands may be either conditional or unconditional. Unconditional commands are executed by all elements regardless of their current state. The unconditional command **clear**[T], for instance, causes all elements to clear the marker bit representing T. Conditional commands are more common. The command

link-type["→"], **on-tail**[T], **off-head**[T] $\Longrightarrow$ **set-head**[T]

would be executed by any link element meeting the conditions on the left hand side of the arrow: if an element represents a link of type "→", the node at its tail bears the marker T, and the node at its head does not bear marker T, then the link will perform the action specified on the right hand side of the arrow, marking the node at its head with T. Using conditional commands, it is possible to address particular nodes by name; the node named x would be selected by placing the restriction **name**[x] on the left hand side of the conditional; only the element representing that node would then respond. This technique is used to select and mark the initial node at the beginning of a marker propagation. Looping is accomplished with a simple **loop** ... **endloop** construct, which repeats the body of the loop until no conditional command contained in the loop can be executed.

To illustrate this notation, we provide a description of the parallel marker propagation algorithm for computing the transitive closure of the "→" relation, starting from a node x. The result of the algorithm, in a net Γ, is to assign the marker T to all nodes y such that Γ supports $x \rightarrow y$.

procedure transclose(x: **node**; T: **marker**) = **begin**
clear[T];
name[x] $\Longrightarrow$ **set**[T];
loop

link-type[“$\rightarrow$”], **on-tail**[T], **off-head**[T] $\Longrightarrow$ **set-head**[T]
endloop
end

Note that a command to propagate T across a link includes the condition that T is not already on the link's head node; this makes it possible for the left hand side of the command to fail, causing the loop to terminate. When none of the conditional commands in a loop body can be performed by any element, the propagation described by the loop is complete.

The two most common and useful parallel marker propagation algorithms for performing inferences over semantic nets are known as the upscan and downscan algorithms. (Their names, due to Fahlman, reflect the primary direction in which markers flow through the graph.) Given a net Γ and a node x, the upscan algorithm can be used to find all the statements $x \rightarrow y$ and $x \not\rightarrow y$ that are supported by Γ. It begins by assigning to x a particular marker—here we use T—and ends when: (i) the marker T has propagated to all those nodes y for which Γ supports $x \rightarrow y$, and (ii) a second marker, F, has propagated to all the nodes z for which Γ supports $x \not\rightarrow z$. (If x were the Jumbo node, for instance, an upscan of x would find all the kinds of which Jumbo is an instance, such as elephant and mammal, as well as the kinds of which Jumbo is definitely not an instance, such as bird.) We present here a version of the upscan algorithm appropriate for monotonic semantic nets. It differs from the nonmonotonic version given by Touretzky in [15], since the absence of exceptions allows contrapositive forms of reasoning, affecting the flow of markers through the network.

procedure upscan(x: **node**; T, F: **marker**) = **begin**
clear[T,F];
name[x] $\Longrightarrow$ **set**[T];
loop
link-type[“$\rightarrow$”], **on-tail**[T], **off-head**[T] $\Longrightarrow$ **set-head**[T];
link-type[“$\rightarrow$”], **on-head**[F], **off-tail**[F] $\Longrightarrow$ **set-tail**[F];
link-type[“$\not\rightarrow$”], **on-tail**[T], **off-head**[F] $\Longrightarrow$ **set-head**[F];
link-type[“$\not\rightarrow$”], **on-head**[T], **off-tail**[F] $\Longrightarrow$ **set-tail**[F];
endloop
end

This algorithm is both correct and complete, in the following sense.

Theorem 4: Let $C \in \mathcal{T}$ be the color assigned to the node y as the result of an upscan of the node x in Γ. Then $\mathrm{T} \in C$ iff $\Gamma \models x \rightarrow y$, and $\mathrm{F} \in C$ iff $\Gamma \models x \not\rightarrow y$.

Proof. The key is to see that upscan will propagate the marker T from x to y in Γ just in case Γ permits a positive path from x to y, along which the marker is actually propagated; and also that an execution of upscan will result in the propagation of the marker F to y just in case Γ permits a negative path from x to y, along which the propagation can take place.

For example, suppose $T \in C$. Then there exists at least one sequence of nodes $u_0 \ldots u_n$ with $x = u_0$ and $y = u_n$ such that on iteration i of the upscan loop a T was propagated from node u_{i-1} to node u_i. Obviously, Γ contains the links $u_{i-1} \rightarrow u_i$, $1 \leq i \leq n$. Therefore Γ permits the path $\langle u_0, \ldots, u_n \rangle$, so $\Gamma \models x \rightarrow y$.

Likewise, suppose $\Gamma \models x \not\rightarrow y$. We know, then, that Γ permits some negative path $\sigma = \langle u_0, \ldots, u_n; v_1, \ldots, v_m \rangle$, with $x = u_0$ and $y = v_m$; assume without loss of generality that σ is a path of minimal length. It is easy to see that after n iterations of the upscan loop, the upscan algorithm will propagate the marker T to the node u_n; after m more iterations, the marker F will be placed on the node v_m. Thus, we will have $F \in C$. The other two cases are similar. ∎

The downscan algorithm is a kind of converse to upscan: it is used to determine all the assertions $x \rightarrow y$ and $x \not\rightarrow y$ supported by a net, given y rather than x. (A downscan of mammal, for example, would find all subtypes and instances of mammal, including elephants and Jumbo, as well as all the kinds and individuals that are definitely not mammals, such as birds and Tweety.) We present here the version appropriate for monotonic nets; again, it is a bit different from the nonmonotonic version of [15].

```
procedure downscan(y: node; T, F: marker) = begin
  clear[T,F];
  name[y]  ⟹  set[T];
  link-type["↛"], on-head[T], off-tail[F]  ⟹  set-tail[F];
  loop
    link-type["→"], on-head[T], off-tail[T]  ⟹  set-tail[T];
    link-type["→"], on-head[F], off-tail[F]  ⟹  set-tail[F];
  endloop
end
```

Note that the downscan loop contains only two propagation commands while the upscan loop requires four. The reason for this is that a T on one end of a "$\not\rightarrow$" link shouldn't propagate an F to the other end during a downscan, except for "$\not\rightarrow$" links whose head node

is the starting node of the downscan. For just these links, an F is put on their tail node by the conditional command that immediately precedes the loop. The correctness and completeness results contained in Theorem 4 can easily be duplicated for this downscan algorithm.

Since all the link elements in a PMPM operate in parallel, and there is no contention when a marker arrives at a node simultaneously from different links, these parallel marker propagation algorithms have the desirable property that their running time depends only on the depth of the graph to which they are applied, not on the total number of nodes or links it contains or the degree (number of ingoing or outgoing links) of any node. In fact, the maximum running times of the uspcan and downscan algorithms are linear in the depth of the inheritance graph. Let k be the number of links in the longest shortest path between any two nodes in the graph, (for negative paths, which are of form $\langle x_0, \ldots, x_n; y_1, \ldots, y_m \rangle$, $k = n + m$); then the maximum running time for a downscan of any node is $2k + 5$ commands, and for an upscan it is $4k + 6$ commands. The downscan algorithm consists of 3 initialization commands plus a loop containing 2 propagation commands. If there is any path between nodes x and y in Γ there must be a path of length at most k, so that after k times through the loop, all nodes that can be marked will have been marked via at least one path. When the loop is repeated one more time, the left hand sides of both conditional propagation commands fail because there are no nodes left to mark, causing termination. Thus, after $3 + 2(k + 1) = 2k + 5$ commands, the algorithm terminates. Since the upscan algorithm consists of 2 initialization commands plus a loop containing 4 propagation commands, its maximum running time is $2 + 4(k + 1) = 4k + 6$ commands.

Scans can be sped up by adding extra links to shorten the shortest paths between nodes. Let k be the maximum number of times the upscan or downscan algorithm executes the body of its main loop given any node of Γ. This number can be reduced to any value from 1 to $k - 1$ by the following method. To reduce the running time to at most j iterations, $1 \leq j < k$, it suffices to put in direct links $x \rightarrow y$ or $x \not\rightarrow y$ between all pairs of nodes x and y such that an upscan of x in Γ marks y with T or F, respectively, after greater than j iterations of the loop. A similar technique can be used to speed up downscans. This simple technique is not optimal in the number of links added, however.

8. Conclusion

We have established in this report a number of close connections between semantic inheritance networks, on the one hand, and more traditional concepts from logic—both proof-theoretic and model-theoretic—on the other. We do not claim that these results are logically profound; but they do succeed, we believe, in showing that fruitful interactions can arise between traditional model-theoretic and proof-theoretic ideas, and the structures that have evolved within artificial intelligence.

Although, in this report, we have remained within a nonmonotonic context, and considered only a very restricted language, we are hopeful that this research will provide a foundation for further work that will be valuable both for logic and artificial intelligence. The extensions that we now envisage fall into two broad categories: developments in expressive power, and developments that provide for the ability to accommodate exceptions.

It is possible to enhance the expressive power of the language analyzed in this report while remaining entirely within a monotonic framework, by adding relational predicates, connectives, and even quantifiers. We believe that many of the results contained here can be generalized in a straightforward way to semantic networks containing hierarchies of n-ary relational predicates. Although such networks may not be realizable on a PMPM, they appear to have efficient inference algorithms on more powerful architectures, such as massively parallel message passing machines [8]. As far as connectives and quantifiers are concerned, the obvious place to start is with the four-valued connectives defined in [2,3]; but it seems that if we are interested in discovering the logical theories that arise naturally from a consideration of semantic nets, we may be forced to adopt a logic with certain constructive features not contained in the four-valued connectives.

The second way of extending this work is to shift to a nonmonotonic context. Here, of course, the problems are much more difficult, though some of the sequent rules presented in this report carry over unchanged to the nonmonotonic case. Still, the different approaches we have mapped out—allowing us, really, to look at the same thing from three different perspectives (semantic nets, model theory, and proof theory)—may provide new leverage for understanding the nature of nonmonotonic inheritance. The possibility of applying proof theory in this way is particularly intriguing, since it is one of the most well-developed areas of logic, and as far as we know it has never been exploited as a technique for analyzing nonmonotonic reasoning.

REFERENCES

[1] Anderson, A. and N. Belnap, *Entailment, Vol. I.* Princeton University Press, Princeton, 1975.

[2] Belnap, N., "How a computer should think." In G. Ryle (ed.), *Contemporary aspects of philosophy,* Oriel Press, 1977, pp. 30–56.

[3] Belnap, N., "A useful four-valued logic." In J.M. Dunn and G. Epstein (eds.), *Modern uses of multiple-valued logic*, D. Reidel, Dordrecht, 1977, pp. 8–37.

[4] Corcoran, J., "Aristotle's natural deduction system." In J. Corcoran, (ed.), *Ancient logic and its modern interpretations*, D. Reidel, Dordrecht, 1974, pp. 85–131.

[5] Ellis, B., *Rational Belief Systems.* APQ Library of Philosophy, Rowman and Littlefield, Totowa, NJ, 1979.

[6] Fahlman, S. E., *NETL: a system for representing and using real-world knowledge.* The MIT Press, Cambridge, 1979.

[7] Hayes, P. J., "The logic of frames." In D. Metzing (ed.), *Frame conceptions and text understanding,* Walter de Gruyter and Co., 1979, pp. 46–61. Reprinted in R. J. Brachman and H. J. Levesque (eds.), *Readings in knowledge representation,* Morgan Kaufman, Los Altos, CA, 1985, pp. 287–297.

[8] Hillis, W. D., *The connection machine.* The MIT Press, Cambridge, 1985.

[9] Kneale, W. and M. Kneale, *The development of logic.* Oxford University Press, Oxford, 1962.

[10] Leblanc, H., 'Alternatives to standard first-order semantics', in D. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic, Vol. I,* Reidel, Dordrecht, 1983, pp. 189–274.

[11] Martins, J. and S. Shapiro, "Theoretical foundations for belief revision." In *Reasoning about knowledge,* J. Halpern (ed.), Morgan Kaufmann Publishers, Los Altos, CA, 1986, pp. 383–398.

[12] Mitchell, J. and M. O'Donnell, "Realizability semantics for error-tolerant logics." In *Reasoning about knowledge,* J. Halpern (ed.), Morgan Kaufmann Publishers, Los Altos, CA, 1986, pp. 363–381.

[13] Shapiro, S. and M. Wand "The relevance of relevance." Technical Report No. 46, Computer Science Department, Indiana University, 1976.

[14] Smiley, T., "What is a syllogism?" *Journal of Philosophical Logic* **2**, 1973, pp. 136–154.

[15] Touretzky, D. S., *The mathematics of inheritance systems.* Morgan Kaufmann, Los Altos, CA, 1986.

[16] Van Fraassen, B. C., "Quantification as an act of mind." *Journal of Philosophical Logic* **11**, 1982, pp. 343–369.