Computer Science Department Technical Report
Cognitive Systems Laboratory
University of California
Los Angeles, CA 90024-1596

INHERITANCE = CHAINING + DEFEAT

Hector Geffner
Tom Verma

June 1989
CSD-890039

Submitted to the 4th International Symposium
on Methodologies for Intelligent Systems.

TECHNICAL REPORT
R-129
March 1989

Inheritance = Chaining + Defeat

Hector Geffner and Tom Verma*
Cognitive Systems Lab.
Dept. of Computer Science, UCLA
LA, CA 90024

Abstract

A simple but powerful scheme for inheritance reasoning is presented. The inheritance scheme embodies clear notions of specificity and defeat, yielding an intuitive, justifiable behavior. We illustrate the scheme with several non-trivial examples and prove its consistency and complexity properties. Finally, a polynomial algorithm which provides a satisfactory sound but incomplete approximation of the inheritance criterion proposed is presented.

1 Introduction

Defeasible inheritance hierarchies constitute appealing devices for organizing prototypical knowledge about classes. Rather than explicitly stating the attributes of each possible individual, individuals are implicitly assumed to inherit a certain set of attributes by virtue of the place they occupy in the hierarchy.

Several systems embedding this type of facility have been reported (e.g. [Fahlman, 79]). An important feature supported by these systems, unlike those based on classical first order logic, has been the ability to accommodate objects belonging to classes with conflicting attributes. In these cases, the systems normally assume the objects to inherit from their closest classes in the hierarchy. This view of inheritance permitted efficient implementations and appeared to correctly embed a preference favoring information attached to more specific classes.

More recently, it has been noted that these simple inheritance strategies are bound to produce anomalous results under a number of circumstances [Etherington and Reiter, 83; Touretzky, 84]. Touretzky, for instance, illustrated some of the anomalies that result from

*This work was supported in part by Grants # N00014-87-K-2029 and IRI-8610155.

the presence of ambiguities and redundancies in the networks. Further work has illustrated other subtle aspects of inheritance reasoning and has revealed a space of options wider than once thought [Touretzky *et al.*, 87]. The inheritance reasoners of Horty *et al.* [87] and Sandewal [86], as well as the more general systems of defeasible inference of Nute [86], Delgrande [87], Loui [87] and Geffner and Pearl [87], have recently explored different alternatives.

Important clues about some of the features a reasonable inheritance scheme is likely to possess have emerged from this body of work. Several of these schemes appear to suggest, for instance, the adequacy of definitions of inheritance cast in inductive form. These schemes regard a defeasible link $p \rightarrow q$ as asserting a context-dependent relation which permits to establish q whenever p represents all the available evidence. The task of a definition of defeasible inheritance then appears to be that of determining from these 'base cases' whether q should be warranted in contexts with evidence other than p. An adequate definition will be such that intuitive conclusions are captured, counterintuitive conclusions are rejected, and meaningful justifications are facilitated.

Horty *et al.* [87] proposed an account of inheritance reasoning along these lines, which they showed to possess some interesting properties. The core of such an account resides in the conditions under which 'supported' paths can be chained with links in the net to yield longer 'supported' paths. The scheme has the virtue of simplicity in its form and the elimination of some idiosyncratic features of a former account reported by Touretzky [86]. However, its reliance on questionable notions of 'preemption' and 'skepticism' results in a behavior which does not always admit clear justification and which often departs from intuition.

In this paper we present an alternative account of inheritance reasoning which, we claim, rests on more solid principles. Our proposal draws on the approach to default inference advocated in [Geffner and Pearl, 87] and results in a system which closely resembles Horty's *et al.* in form, but which departs from it in some significant ways. In particular, we argue that it embodies a more appropriate notion of specificity and defeat, a safer approach towards ambiguity and a capability to deal with cyclic networks.

The paper is organized as follows. In section 2 we describe the proposed inheritance scheme, whose main logical properties are summarized in section 3. In section 4 we analyze a number of examples illustrating its main features and how it departs from related accounts. In section 5 we analyze the complexity of the inheritance scheme proposed, which turns out to be intractable, and introduce a polynomial algorithm which, we claim, constitutes a satisfactory sound but incomplete approximation. The main results and possible extensions are discussed in section 6.

2 A Scheme for Defeasible Inheritance

2.1 Notation

Every inheritance network is associated with a set of positive and negative paths or arguments. Positive paths are sequences of the form $p \cdot q \cdot r \cdots$ whose members are propositional symbols. Negative paths are positive sequences followed by a single negated propositional symbol, e.g. $p \cdot q \cdot \neg r$.

Normally, some propositional symbols are taken to stand for classes while others are taken to stand for individuals. Here, we shall not need to be concerned with such a distinction so, without loss of generality and unless stated otherwise, we shall assume all propositional symbols to stand for classes.

Small greek letters $\sigma, \tau, \ldots$, are used as variables ranging over (potentially empty) path sequences, and propositional letters from the end of the alphabet, x, y, ..., are exclusively used as variables ranging over literals, i.e. positive and negated propositional symbols. Furthermore, we use the notation $\sim x$ for complementation, so that if x stands for p, $\sim x$ denotes $\neg p$; and if x stands for $\neg p$, $\sim x$ denotes p.

Path sequences of length two are referred as links or defaults. A path $\sigma \cdot x$, can be chained with a link $x \cdot y$ to yield a new path $\sigma \cdot x \cdot y$. An inheritance net Γ is a set of links whose antecedents are positive literals. We will use Γ^* to refer to the set of paths comprising those in Γ together with those which can be obtained from Γ by successive chaining.

2.2 The Rules of Inference

In order to characterize the literals that follow from a given propositional symbol, we introduce the consequence operator '$\mid\!\!\sim_\Gamma$.' For a propositional symbol x and a literal y, the notation $x \mid\!\!\sim_\Gamma y$ states that y is derivable from x in the network represented by Γ. Often, however, we simply write $x \mid\!\!\sim y$, leaving Γ implicit. Such notation will mean that x has the property y, if x stands for an individual, and that typically, xs are ys, if x is a class. Defining inheritance in terms of a binary relation over literals rather that in terms of paths has its benefits which will become apparent later.

The consequence relation '$\mid\!\!\sim_\Gamma$' is essentially characterized by two inference rules. The first rule:

Rule 1 (Links)
If $x \cdot y \in \Gamma$, then $x \mid\!\!\sim_\Gamma y$

simply states that y follows from x if there is a link connecting x to y in the net. Similar

rules appear in [Horty *et al.*, 87; Delgrande, 87; Geffner and Pearl, 87]. Such a rule expresses the view that a default 'if x then y' can be interpreted as a license to infer y, when x represents all the available evidence.

The second rule spells out the conditions under which a path and a link can be chained together to yield a conclusion only implicit in the net. A safe rule to that effect would be the following:

Rule 2' (Conservative Chaining)
If $x \mathrel{|\!\!\sim}_\Gamma y$, $y \cdot z \in \Gamma$ and there is no path $x \cdot \sigma \cdot \sim z \in \Gamma^*$, then $x \mathrel{|\!\!\sim}_\Gamma z$.

That is, chaining would be allowed in the absence of arguments against the new consequent. This restriction, however, turns out to be too strong. Often, many of the defaults that make up the net can be assumed *not to be applicable* to a given class. For instance, a default link representing a relation such as 'birds fly' is supposed not to be applicable to 'penguins.' Thus, in trying to determine what properties penguins can be inferred to possess, those links or, more generally, those paths, can be safely ignored. We shall define below, a condition for path or argument *defeat,* allowing us to identify the paths that can be assumed to be non-applicable to a given class or individual x. These will turn out to be the paths *defeated* by x. Rule 2 then becomes:

Rule 2 (Chaining)
If $x \mathrel{|\!\!\sim}_\Gamma y$, $y \cdot z \in \Gamma$ and every path $x \cdot \sigma \cdot \sim z \in \Gamma^*$ is **defeated** by x, then $x \mathrel{|\!\!\sim}_\Gamma z$.

This rule states that the link $y \cdot z$ can be chained to yield a new conclusion z about x, whenever all the arguments against z are defeated by x. For easy reference, we will use the name **I** to refer to the inheritance account that results from this pair of rules and the definition of defeat to be introduced below.

2.3 Defeat

Defeat is defined in terms of a *dominance* relationship, which in turn, is defined in terms of the derivability relation ' $\mathrel{|\!\!\sim}_\Gamma$ ' introduced above.

> **Definition (Dominance).** A link $x \cdot y$ in Γ is said to **dominate** a path $z \cdot \sigma \cdot \sim y$, if $x \mathrel{|\!\!\sim}_\Gamma z$. In such case, we also say that the path is dominated by x.

For instance, from the net depicted in fig. 1 we can infer that the link $\texttt{penguin} \cdot \neg\texttt{fly}$ dominates the link $\texttt{bird} \cdot \texttt{fly}$. This is a consequence of the definition above and rule 1, which permits us to infer `bird` from `penguin`. As a result, we can also say that the link $\texttt{bird} \cdot \texttt{fly}$ is dominated by the `penguin` class. This will imply that, while the link $\texttt{bird} \cdot \texttt{fly}$ is applicable to `birds`, it will not be applicable to `penguins`. To that effect, path defeat is defined as follows:

Definition (Defeat). A path $\sigma \cdot \tau \cdot \upsilon$ is **defeated** by a class or individual x, if x or a consequence w of x, dominates τ.

For the example above, thus, we obtain that paths containing the link `bird` $\cdot$ `fly` will be defeated by `penguin` and by any other class or individual whose 'penguiness' can be established.

Example. Consider an inheritance network Γ given by the following relationships: 'things do not fly,' 'birds fly,' 'penguins do not fly,' 'penguins are birds,' 'bird are things,' and 'Tweety is both a bird and a penguin' (fig. 1). From such a network we can show 'Tweety does not fly' by means of the following derivation:

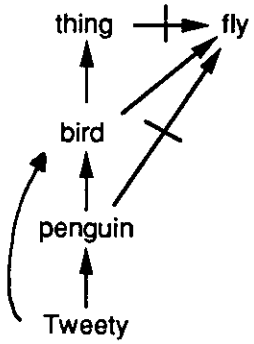

Figure 1: A Simple Inheritance Network

1. `Tweety` $\vdash$ `penguin` ; Links
2. `penguin` $\vdash$ `bird` ; Links
3. `Tweety` $\vdash \neg$`fly` ; Chaining, line 1, `penguin` $\cdot \neg$`fly` $\in \Gamma, \ldots$

where the chaining is justified by the fact that `Tweety` defeats the conflicting paths `Tweety` $\cdot$ `bird` $\cdot$ `fly` and `Tweety` $\cdot$ `penguin` $\cdot$ `bird` $\cdot$ `fly`, as a result of being a member of the `penguin` class (step 1); a class that dominates the link `bird` $\cdot$ `fly`.

Notice that, by a similar argument – the dominance of `bird` over the link `thing` $\cdot \neg$`fly` – it can be shown that `Tweety` defeats the link connecting `thing` to $\neg$`fly` as well. Indeed, the links `bird` $\cdot$ `fly` and `thing` $\cdot \neg$`fly` constitute the minimal paths defeated by `Tweety`. The removal of these links from Γ result into a new net Γ' which can be usefully regarded as the net Γ as 'viewed' from `Tweety`. Interestingly, it can be proven that whatever properties `Tweety` can be shown to inherit in Γ by means of rules 1 and 2, are the same properties that `Tweety` can be shown to inherit in Γ', by means of rules 1 and 2' (conservative chaining).

It should be pointed out that it is not the common view to regard a path such as `Tweety` $\cdot$ `bird` $\cdot$ `thing` $\cdot \neg$`fly` as defeated, when all its associated properties are true of `Tweety`. Horty *et al.* for instance, would regard such path as supported. Our approach, however, is based on viewing defeat as a *monotonic* extension of the dominance relation. *That is, if a default 'things do not fly' is not applicable to* `birds`*, then it is assumed not to*

be applicable to any type of **bird**; **penguins** *and* **Tweety** *included.* The resulting behavior remains in agreement with intuition, but its justification, we believe, becomes clearer.

We shall refer to paths of the form $x \cdot x_1 \cdots x_n$, $n \geq 1$, where each x_i is derivable from x, as *valid* paths. However, only those valid paths which are not defeated by x will be said to be *certified.* We have seen above an example of a valid but non-certified path. In the next section, a correspondence between classes y derivable from x and the existence of certified paths connecting x to y will be established.

3 Properties

In this section, we summarize some of the salient properties of the inheritance scheme proposed. Proofs are in the appendix; most proceed by induction on the length of the derivations.[1]

Definition. A network Γ is said to be **consistent**, if there is no pair of propositions x and y such that $x \mathrel{\mid\!\!\sim_\Gamma} y$ and $x \mathrel{\mid\!\!\sim_\Gamma} \sim y$.

Definition. A network Γ is said to be **safe**, if for every pair of links $x \cdot y$ and $z \cdot \sim y$ in Γ, x and z are distinct and there is no cyclic path in Γ embracing both x and z.

A cyclic path that embraces x and z is a path of the form $\cdots x \cdots z \cdots x \cdots$.

Theorem (Consistency)
If Γ is safe, then Γ is consistent.

Interestingly, a similar sufficient and, indeed, necessary condition for consistency, appears in Pearl's account of defeasible inheritance in the context of a probabilistic semantics ([Pearl, 88], Chap. 10). Most work on inheritance reasoning, however, has concentrated on acyclic networks, in which even negative cycles are ruled out. This theorem indicates, however, that no inconsistencies originate from negative cycles.

The following result assumes Γ to represent a safe network and it establishes a correspondence between derivations and certified paths. Let us recall that a certified path is a path of the form $x \cdot x_1 \cdots x_n$ not defeated by x, such that each x_i follows from x.

Theorem (Correspondence)
$x \mathrel{\mid\!\!\sim_\Gamma} y$ if and only if there is a certified path $x \cdot \sigma \cdot y$ in Γ^*.

[1] For a precise definition of "derivation", see section 4.

It follows, then, that every conclusion has an associated undefeated path. Thus, defeated paths can be consistently ignored both as potential sources of conflict and as potential sources of support. This captures the intuition that the virtual network a class or individual effectively 'sees' excludes the paths it defeats.

Both theorems suggest that the notion of defeat introduced does not lead to pathological behavior in the family of safe networks. In the next section we will further illustrate that the behavior legitimized by **I** remains in agreement with intuition and accepts simple justifications.

4 Examples

Derivations. In order to provide clear justifications for derivations we will find useful to replace the chaining rule by an equivalent set of three rules which make explicit mention of dominance and defeat relations. For that purpose we will use the notation $\langle y|x|z\rangle$ to denote that x dominates the paths between y and z.[2] Triplets of the form $\langle y|x|z\rangle$ will be referred as *domination triplets*. On the other hand, triplets of the form $\langle y\|x\|z\rangle$ constitute *defeat triplets* and are to be understood as asserting that x defeats the paths of the form $\sigma_1 \cdot y \cdot \sigma_2 \cdot z \cdot \sigma_3$, for arbitrary, potentially empty subpaths σ_i. In words, that those paths are not applicable to x.

With slight abuse of notation, the original pair of rules can then be rewritten as:

Rule 1 (Links)
If $x \cdot y \in \Gamma$, then $x \mathrel{\overset{\Gamma}{\models}} y$

Rule 2.1 (Dominance)
If $x \mathrel{\overset{\Gamma}{\models}} y$ and $x \cdot \sim z \in \Gamma$, then $\mathrel{\overset{\Gamma}{\models}} \langle y|x|z\rangle$

Rule 2.2 (Defeat)
If $x \mathrel{\overset{\Gamma}{\models}} w$ or $x = w$, and $\mathrel{\overset{\Gamma}{\models}} \langle y|w|z\rangle$, then $\mathrel{\overset{\Gamma}{\models}} \langle y\|x\|z\rangle$

Rule 2.3 (Chaining)
If $x \mathrel{\overset{\Gamma}{\models}} y$, $y \cdot z \in \Gamma$ and for every path $x \cdot \sigma \cdot \sim z \in \Gamma^*$,
$x \cdot \sigma \cdot \sim z = \sigma_1 \cdot w_1 \cdot \sigma_2 \cdot w_2 \cdot \sigma_3$ and $\mathrel{\overset{\Gamma}{\models}} \langle w_1\|x\|w_2\rangle$, then $x \mathrel{\overset{\Gamma}{\models}} z$.

Proofs, hereafter, will proceed according to this set of rules.

[2]The form of such notation is motivated by the fact that by dominating those paths x *separates* y from z in the corresponding inheritance graph. A similar notation is used in the context of probabilistic networks, where a graph separation criterion is used for capturing the set of conditional independence assumptions embedded in a network (see [Pearl, 88], Chap. 3).

Example 1: No coupling. We first consider the network $\Gamma = \Gamma_1$ depicted in fig. 2. The derivation for $A \mathrel{|\!\sim}_\Gamma F$ proceeds as follows:

1. $A \mathrel{|\!\sim}_\Gamma B$; Links
2. $A \mathrel{|\!\sim}_\Gamma C$; Chaining 1, $B \cdot C \in \Gamma_1$
3. $\mathrel{|\!\sim}_\Gamma \langle B \| A \| D \rangle$; Dominance + Defeat 1, $A \cdot \neg D \in \Gamma_1$
4. $A \mathrel{|\!\sim}_\Gamma F$; Chaining 2, 3, $C \cdot F \in \Gamma_1$

The derivation of F rests on the assumption that the link $B \cdot D$ is not applicable to A and can therefore be ignored. Note that B, the only parent of A, cannot be shown to inherit F. In terms of Touretzky *et al.* [87], thus, there is no coupling between the properties of a class and the properties of its superclasses. Inheritable properties thus cannot be 'visualized' as flowing downwards from parents to sons.

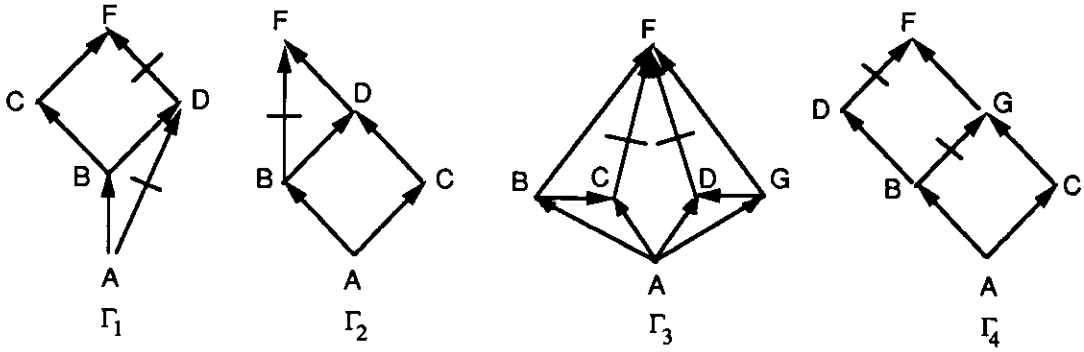

Figure 2: $\Gamma_1 - \Gamma_4$

Example 2: Path preemption. In $\Gamma = \Gamma_2$ (fig. 2), in a similar way, A can be shown to inherit $\neg F$ as follows:

1. $A \mathrel{|\!\sim}_\Gamma B$; Links
2. $B \mathrel{|\!\sim}_\Gamma D$; Links
3. $\mathrel{|\!\sim}_\Gamma \langle D | B | F \rangle$; Dominance 2, $B \cdot \neg F \in \Gamma_2$
4. $\mathrel{|\!\sim}_\Gamma \langle D \| A \| F \rangle$; Defeat 1, 3
5. $A \mathrel{|\!\sim}_\Gamma \neg F$; Chaining 1, 4, $B \cdot \neg \in \Gamma_2$

Touretzky *et al.* refer to this net as an example of off-path preemption, as the path $A \cdot C \cdot D \cdot F$ is preempted by a literal B which does not lie along the path.

Example 3: No best path. The next example illustrates that it is not necessary for a single path to be strictly 'superior' to all competing paths in order to be certified. Indeed, in $\Gamma = \Gamma_3$ (fig. 2), neither $A \cdot G \cdot F$ is 'superior' to $A \cdot C \cdot \neg F$, nor $A \cdot B \cdot F$ is 'superior' to $A \cdot D \cdot \neg F$. Still, as the following derivation shows, the positive paths prevail over the negative paths:

1. $A \mathrel{|\!\sim_\Gamma} B$; Links
2. $A \mathrel{|\!\sim_\Gamma} G$; Links
3. $B \mathrel{|\!\sim_\Gamma} C$; Links
4. $\mathrel{|\!\sim_\Gamma} \langle C|B|\neg F\rangle$; Dominance 3, $B \cdot F \in \Gamma_3$
5. $G \mathrel{|\!\sim_\Gamma} D$; Links
6. $\mathrel{|\!\sim_\Gamma} \langle D|G|\neg F\rangle$; Dominance 5, $G \cdot F \in \Gamma_3$
7. $\mathrel{|\!\sim_\Gamma} \langle C\|A\|\neg F\rangle$; Defeat 1, 4
8. $\mathrel{|\!\sim_\Gamma} \langle D\|A\|\neg F\rangle$; Defeat 1, 6
9. $A \mathrel{|\!\sim_\Gamma} F$; Chaining 1, 7, 8, $B \cdot F \in \Gamma_3$

Examples 4: Propagation of ambiguity. The conclusions warranted in the examples above are all in agreement with those which would be obtained by Horty's *et al.* scheme. We will consider now some examples in which the behavior resulting from both schemes differ. The first such an example, from [Touretzky *et al.*, 87], is represented by the cascaded diamonds of Γ_4 (fig. 2). In the 'skeptical' scheme of Horty *et al.*, paths of the form $x \cdot \sigma \cdot z$, where σ comprises classes which x cannot be shown to inherit, are ignored. As a result, in Γ_4, the path $A \cdot B \cdot D \cdot \neg F$ is left unchallenged and A inherits $\neg F$. In our scheme, for A to inherit $\neg F$ in the presence of the conflicting path $A \cdot C \cdot G \cdot F$, it is necessary to show that A defeats such path. Since such defeat cannot be derived, no conclusion about F follows.

Examples 5 – 8: Specificity and Cumulativity. Figure 3 depicts a series of networks $\Gamma_5 - \Gamma_8$. In each network we are interested in determining whether a inherits property F, a proposition which we abbreviate as $F(a)$. An aspect common to all $\Gamma_5 - \Gamma_8$ is that a defeats the path $C \cdot G \cdot \neg F$ by virtue of the dominance of B over the latter. As a result, it turns out that we can prove both $a \mathrel{|\!\sim_{\Gamma_5}} F$ and $a \mathrel{|\!\sim_{\Gamma_6}} F$. In Γ_7 and Γ_8 on the other hand, the presence of the undefeated path $a \cdot G \cdot \neg F$ prevents the same conclusion from being certified. It is interesting to note that the scheme of Horty *et al.* would authorize such a conclusion in all these networks: in Γ_5 and Γ_6, on the basis that the parent G of $\neg F$ is not derivable, and in Γ_7 and Γ_8 due to the presence of a 'preemptive' supported path $a \cdot B \cdot C \cdot G$.

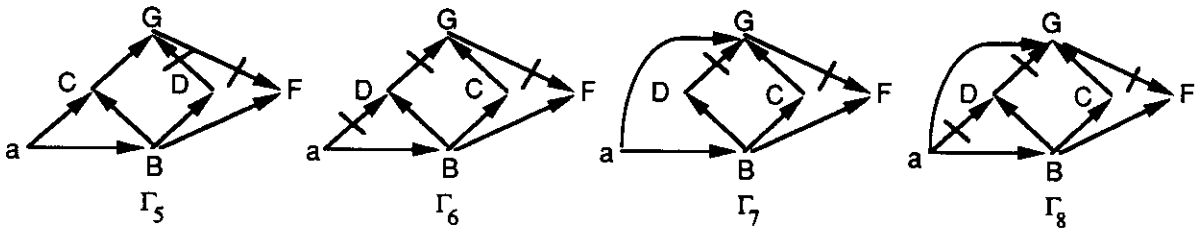

Figure 3: $\Gamma_5 - \Gamma_8$

The different treatment of networks such as Γ_7 and Γ_8 illustrates different views on the conditions that make an argument prevail over competing arguments. Indeed, it is possible to show in our scheme that an individual[3] member of a subclass x of y has the property

[3]For our purposes, any literal without incoming links.

$\sim y$, only when it is also member of a subclass w of $\sim y$, *more specific* than x. Formally:

Theorem (Specificity)
If $a \cdot x \cdot y \in \Gamma^*$ and $a \mathrel{|\!\!\sim}_\Gamma \sim y$, then there must be a link $w \cdot \sim y$ in Γ, such that $a \mathrel{|\!\!\sim}_\Gamma w$ and $w \mathrel{|\!\!\sim}_\Gamma x$.

In the context of Γ_8, this property states that a could be shown to inherit F only if B could be shown to be a subclass of G. Since the latter does not follow, the former does not obtain. In Horty's *et al.* scheme, the appeal to a notion of preemption which depends on additional characteristics of the particular target individual (a, in this case) does not allow such a clear cut notion of specificity to emerge. It should be pointed out, however, that a slight change in their definitions would suffice for that purpose.

The networks depicted in fig. 3 permit to illustrate an additional distinction between Horty's *et al.* scheme and **I**. This is about a property which has been called *cumulativity* by Makinson [89] and *stability* by Horty *et al.* . In the context of inheritance hierarchies, this property establishes that the predicates that an individual a can be shown to inherit in a network Γ', formed by adding a link $a \cdot x$ to Γ, for a class x such that $a \mathrel{|\!\!\sim}_\Gamma x$, are the same as those predicates that a can be shown to inherit in the original network Γ. This property holds in the scheme of Horty *et al.* and in systems of defeasible inference such as [Geffner and Pearl, 87]. As we show below, however, it does not hold in **I**.

Consider the net Γ_6 in fig. 3. In such a network, it is simple to show that a inherits both G and F. Cumulativity then implies that a should still inherit F when a link $a \cdot G$ is added to Γ_6. The resulting network, however, is Γ_8, where we have shown that a does not inherit F.

Thus, it turns out that any scheme must reject either cumulativity, $a \mathrel{|\!\!\sim}_{\Gamma_6} G$ or must be capable of deriving $a \mathrel{|\!\!\sim}_{\Gamma_8} F$. We have chosen to reject cumulativity. Indeed, we are inclined to believe that cumulativity is not necessarily a property of 'correct' defeasible inference but a property about *belief dynamics*, namely a 'rational' policy of belief revision in the light of new information. In other words, we argue that in a context such as Γ_6, preserving the belief $F(a)$ upon *confirming* $G(a)$, is the reasonable thing to do. Not as reasonable however, would be to switch to $F(a)$ upon learning $\neg D(a)$ in the context of Γ_7. The system **I** cannot make such a distinction and, thus, remains uncommitted in both cases.

Example 9: Cycles. In the last example we consider the network Γ_9 depicted in figure 4. Unlike the type of inheritance networks dealt with in the literature, Γ_9 contains a (negative) cycle. Such net could represent, for instance, that 'most university students are around their twenties', that 'people in their twenties are adults' and that 'most adults are not in their twenties.'

The derivation for $a \mathrel{|\!\!\sim}_{\Gamma_9} C$ rests on the fact that B dominates the link $D \cdot \neg C$ and thus, a can be shown to defeat the conflicting path $a \cdot D \cdot \neg C$.

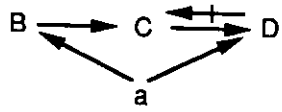

Figure 4: Γ_9: A network with negative cycles

5 Complexity and Implementation

In this section we analyze the complexity of the inheritance scheme proposed, which turns out to be NP-Complete, and propose a polynomial sound but incomplete algorithm which we claim provides a practically satisfactory approximation.

The main result is summarized by the following theorem which is proved in the appendix:

Theorem (Complexity).
The problem of determining whether $x \mid\!\!\sim_{\Gamma} y$ for an arbitrary Γ is NP-Complete.

The proof[4] relies on a reduction to 3SAT [Garey and Johnson, 79]: for an arbitrary three-satisfiability problem P, a network Γ_P is constructed in which an individual x inherits a property y if and only if P is not satisfiable. In particular, Γ_P is such that an undefeated path connecting x to $\sim y$ exists if and only if P is satisfiable.

We would like to point out that we do not regard the intractability of an inheritance scheme as bearing on its correctness. Complexity and correctness are different matters; propositional entailment is also intractable and hardly anyone would dare to argue, on that basis, for a different interpretation of the standard connectives. A similar situation, we are inclined to believe, arises in the context of inheritance reasoning. The possibility of a 'correct', general and tractable account of defeasible inheritance seems unlikely to us. The tractability of Horty *et al.* for instance, comes at the price of a highly questionable notion of 'skepticism.' In the the rest of this section a different compromise is pursued.

5.1 A Sound Algorithm for Inheritance Reasoning

In this subsection we present a polynomial algorithm which computes some, but not *all* of the inferences sanctioned by **I**. Nonetheless, the resulting behavior appears to be quite satisfactory. It is indeed sound and it correctly handles the examples mentioned in the literature, including those in the previous section. Figure 5 depicts what we believe is the

[4]The proof is a variation of a technique used by Selman [89] in assessing the complexity of Touretzky's [86] inheritance scheme.

simplest type of network in which the algorithm fails to draw a conclusion – $a \not\vdash_{\Gamma} F$ – which is authorized by **I**.[5]

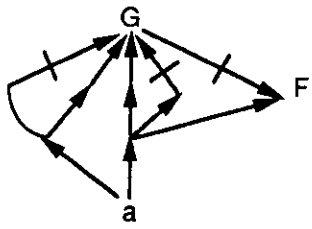

Figure 5: An example of a net where the algorithm does not match the power of **I**

The algorithm computes in polynomial time and space the set of literals that follow from each positive literal in the net. In other words, it computes a table from which the answer to any query can be retrieved. Obviously, the key parameter is not retrieval time but the time it takes to construct the table.[6] The computation proceeds by iterations (see appendix). In each iteration i, the algorithm visits each positive literal x in the net and attempts to extend its set of consequences $C^{i-1}(x)$ by incrementally computing $\Delta C^i(x)$. This is accomplished by chaining, i.e. literals y with a non-empty set of parents $P(y)$ in $C^{i-1}(x) \cup \Delta C^i(x)$ are selected one by one, and the existence of undefeated paths connecting x to $\sim y$ is tested. This test is implemented by recursively spreading markers from x to its neighbor literals, which are absorbed only by literals whose complements are in $C^{i-1}(x) \cup \Delta C^i(x)$, or by literals in $P(y)$ or one of their consequences. It can be shown that the lack of a mark in $\sim y$ at the end of this propagation process is a sound indication of the absence of undefeated paths between x and $\sim y$.[7] If so, y can be added to $\Delta C^i(x)$, and other possible consequences of x can be tried. Otherwise, the next attempt to derive y from x is postponed until the next iteration.

Note that each link in the net is 'tried' at most once in each iteration. Taking into account the complexity of the marker propagation process, we obtain that the complexity of each iteration is in the order of L^2, where L is the number of links in the net. The fixed point is obtained when an iteration fails to generate a new table entry. Since the table that corresponds to a consistent net with N positive literals contains at most $N * N$ filled entries, no more that N^2 iterations can occur. In practice, however, the fixed point is obtained in a few iterations. All the networks considered so far converge to the fixed point in two iterations, independently of the order in which literals are selected. Inferences which require a large number of iterations correspond to chains of highly embedded derivations, where a proof requires a defeat triplet which in turn requires another defeat triplet and so on. Indeed, we claim that inferences which require a number of iterations, say, greater than four, will likely not deserve the title of commonsense inferences at all. Thus, in practice, the complexity of the algorithm will grow in proportion to the square of the number of

[5] Note that **I** does not draw any conclusions regarding G.

[6] In practice, however, it makes sense not to maintain a complete table but a partial one, where unfilled entries can be computed sufficiently fast.

[7] The presence of a marker in $\sim y$, however, does not necessarily indicate the presence of an undefeated path between x and $\sim y$. This is the source of incompleteness.

links in the net.

This algorithm has been implemented in Common Lisp and a listing appears in the appendix. A sample run is shown below.

```
> (define-net clyde '((clyde african elephant gray)(clyde royal -gray)
                      (royal elephant)))

ROYAL -> ELEPHANT ; ROYAL -> -GRAY ;
ELEPHANT -> GRAY ;
AFRICAN -> ELEPHANT ;
CLYDE -> ROYAL ; CLYDE -> AFRICAN ;
NIL

> (run clyde)


-*-*- Iteration 1 -*-*-

New consequences from ROYAL: -GRAY  ELEPHANT
New consequences from ELEPHANT: GRAY
New consequences from AFRICAN: GRAY  ELEPHANT
New consequences from CLYDE: -GRAY  ELEPHANT  AFRICAN  ROYAL


-*-*- Iteration 2 -*-*-

DONE
```

5.2 What does the algorithm compute?

The algorithm discussed is sound but incomplete. We claim, however, that it is sufficiently powerful to account for most of the reasonable inferences we want to draw from a net. The question arises as whether it is possible to provide an abstract characterization of what the algorithm computes and shed some light into the type of sound conclusions which escape its machinery. It turns out that the algorithm provides a sound and *complete* implementation of the slightly less powerful definition of inheritance given by the following rules:

Rule W.1
If $x \cdot y \in \Gamma$, then $x \Vdash_\Gamma y$

Rule W.2
If $x \Vdash_\Gamma y$, $y \cdot z \in \Gamma$ and for every path $x \cdot \sigma \cdot \sim z \in \Gamma^*$, $x \cdot \sigma \cdot \sim z = \sigma_1 \cdot w_1 \cdot \sigma_2 \cdot w_2 \cdot \sigma_3$,

either $x \mid\!\!\sim_{\Gamma} \neg w_1$ or for a **parent** w of z, $x \mid\!\!\sim_{\Gamma} w$ and w dominates the path $w_1 \cdot \sigma_2 \cdot w_2$, then $x \mid\!\!\sim_{\Gamma} z$

where the dominance relation is to be understood in terms of ' $\mid\!\!\sim_{\Gamma}$ '. This definition generates a subset of the conclusion of **I**: the additional condition $x \mid\!\!\sim_{\Gamma} \neg w_1$ still implies that all paths from x to w_1 have to be defeated, but defeat is now defined as an exclusive prerogative of the parents of z derivable from x. The network depicted in fig. 5 is an example of a situation where such additional condition precludes a derivation which is authorized by ' $\mid\!\!\sim_{\Gamma}$ '.

6 Discussion

We have proposed a simple but powerful scheme for inheritance reasoning. The inheritance scheme embodies clear notions of specificity and defeat, which we have illustrated to yield an intuitive, justifiable behavior in a number of examples.

Inheritance is viewed as a restricted form of chaining. A link connecting a consequence y of x to a proposition z, permits us to infer z from x, when all the paths connecting x to $\sim z$ are defeated by x. Defeat is defined in such a way that a class x defeats those all paths dominated by some of its superclasses. The dominance relation itself, under different disguises, has constituted a basic block of most accounts of multiple inheritance with exceptions since [Touretzky, 84]. The novelty of the proposed scheme is the introduction of a notion of defeat which monotonically extends the dominance relation and which renders a stronger behavior, in closer correspondence with intuition.

The system resembles Horty's *et al.* in form, while departing along some important dimensions. In particular, we have argued that it embeds a different notion of specificity, a safer approach towards ambiguity, and a capability for dealing with cyclic networks.

We have shown the inheritance scheme to be NP-Complete. A polynomial algorithm which provides a satisfactory sound but incomplete approximation has also been presented.

The language of networks considered in this paper is quite limited. There seems to be, however, two extensions that could be accommodated in the proposed scheme and, in particular, within the scope of the algorithm presented. One is the inclusion of conjunctive default links: links with multiple positive literals in the antecedent. The second, is the inclusion of simple strict (undefeasible) links: links which guarantee chaining both forward and backward (contrapositive). These extensions will be reported elsewhere.[8]

[8] A third extension, which is not so simple to integrate within the scope of the algorithm presented, is reasoning by cases. That is, if a conclusion follows when an individual has a property p and also when it has the property $\neg p$, then the conclusion should also follow when the individual is not known to have either p or $\neg p$. Indeed, it is possible to construct examples where reasonable conclusions fail to be sanctioned by **I** due to its inability to reason by cases.

Acknowledgments

We would like to thank Judea Pearl for useful discussions about these and related topics.

References

[Delgrande, 87] J. Delgrande. An approach to default reasoning based on a first-order conditional logic. *Proceedings AAAI-87*, Seattle, 1987.

[Etherington and Reiter, 83] D. Etherington and R. Reiter. On inheritance hierarchies with exceptions. *Proceedings of the AAAI-83*, 1983, pp 104-108.

[Fahlman, 79] S. Fahlman. *NETL: A system for representing and using real-world knowledge.* MIT Press, Cambridge, MA, 1979.

[Gabow *et al.*, 76] H. Gabow, S. Maheshari and L. Osterweil. On two problems in the generation of program test paths. *IEEE Transaction on Software Engineering*, 1976, pp 227-231.

[Garey and Johnson, 79] M. Garey and D. Johnson, *Computers and intractability. A guide to the theory of NP-Completeness.* New York, W. H. Freeman, 1979.

[Geffner and Pearl, 87] H. Geffner and J. Pearl. A framework for reasoning with defaults. TR-94b, October 1987, Cognitive Systems Lab., UCLA. Also to appear in the *Proceedings of the 1988 Meeting of the Society for Exact Philosophy*, Rochester, N.Y., June 1988.

[Horty *et al.*, 87] J. Horty, R. Thomason and D. Touretzky. A skeptical theory of inheritance. *Proceedings AAAI-87*, Seattle, Washington, pp 358-363.

[Loui, 87] R. Loui. Defeat among arguments: a system of defeasible inference. *Computational Intelligence*, 1987.

[Makinson, 89] D. Makinson. General theory of cumulative inference. *Proceedings 2nd Int. Workshop on Non-monotonic Reasoning*, Springer Lecture Notes, January 1989.

[Nute, 86] D. Nute. LDR: a logic for defeasible reasoning. ACMC Research Report 01-0013, University of Georgia, Athens, 1986.

[Pearl, 88b] Pearl J., *Probabilistic reasoning in intelligent systems*, Morgan Kaufmann, Los Altos, 1988.

[Sandewal, 86] E. Sandewal. Non-monotonic inference rules for multiple inheritance with exceptions. *Proceedings of the IEEE*, vol 74, 1986.

[Selman, 89] B. Selman. The tractability of path-based inheritance. *Proceeding of the Workshop on Formal Aspects of Semantics Networks*, Catalina, California, February 1989.

[Touretzky, 86] D. Touretzky. *The mathematics of inheritance systems.* Morgan Kaufmann, 1986.

[Touretzky *et al.*, 87] D. Touretzky, J. Horty and R. Thomason. A Clash of Intuitions: The current state of non-monotonic multiple inheritance systems. *Proceedings of the IJCAI-87*, Milano, Italy, 1987.

A Proofs of Main Theorems

In order to prove the consistency result we will need some lemmas and definitions. All the lemmas below assume an inheritance network Γ with literals among x, y, ... The notation $x * y$ is used to indicate the existence of a path connecting x to y in Γ^*.

The first two lemmas are natural consequences of the inductive definition of '$\mid\!\!\sim_\Gamma$':

Lemma 1 If $x \mid\!\!\sim_\Gamma y$ then $x * y$.

Lemma 2 If $x \mid\!\!\sim_\Gamma y$ then all the paths connecting x to $\sim y$ are defeated by x.

The third lemma follows from the definition of defeat.

Lemma 3 If $x \cdots y \cdot z \in \Gamma^*$ and x defeats all the paths connecting x to z, then either x defeats all the paths connecting x to y or there must be a link $w \cdot \sim z$ in Γ such that $x \mid\!\!\sim_\Gamma w$ and $w * y$.

Indeed, if a path $x_0 \cdots x_n$ is not defeated, but the path $x_0 \cdots x_n \cdot x_{n+1}$ is, there must be a literal w which follows from x_0 and which dominates a path $x_i \cdots x_{n+1}$. That is, w is linked to $\sim x_{n+1}$ and $w \mid\!\!\sim_\Gamma x_i$, for some $i \in [1, n]$. By lemma 1 we get $w * x_i$ and $w * x_n$, from which lemma 3 follows.

Lemma 4 Let $x \cdot \sigma$ be a path in Γ^* and let y be the literal closest to x along σ, such that all the paths from x to y are defeated by x. Then $x \mid\!\!\sim_\Gamma \sim y$.

If there is such literal y, it follows from lemma 3 that there must be a literal w linked to $\sim y$, such that $x \mid\!\!\sim_\Gamma w$. The conclusion $x \mid\!\!\sim_\Gamma \sim y$ is, thus, a consequence of rule 2 and the fact that all the paths from x to y are defeated by x.

Lemma 5 Let $x \cdot \sigma \cdot y$ be a path in Γ^* such that x defeats all the paths connecting x to y. Then, either $x \mid\!\!\sim_\Gamma \sim y$ or there is literal w in σ such that $x \mid\!\!\sim_\Gamma \sim w$.

If all the paths to y are defeated, either y is the closest such literal to x along $\sigma \cdot y$ or a literal w in σ is. The result then is a direct corollary of lemma 4.

Definition 1 Let $D^i_{x,y}$ be a derivation of a conclusion $x \mid\!\!\sim_\Gamma y$ according to rules 1,2.1–4 in section 4. We say that $D^i_{x,y}$ *supports a literal* w, if there is a line $x \mid\!\!\sim_\Gamma w$ in $D^i_{x,y}$. We also say that the derivation $D^i_{x,y}$ *supports a link* $w \cdot z$, if $w \cdot z$ is in Γ and both w and z are

supported by $D^i_{x,y}$.

Clearly, the set of links supported by a derivation $D^i_{x,y}$ must cover a path connecting x to y. On the other hand, not all the links supported are really relevant to the derivation. We will find it useful to isolate among the supported links, the set of *basic* supported links as follows.

Definition 2 A basic supported link is a supported link $y \cdot z$ such that for every other supported link $w \cdot z$ with $w * y$, we also have $y * w$.

Very roughly, the basic supported links which correspond to a derivation $D^i_{x,y}$ are the 'most specific' links supported by $D^i_{x,y}$. Note that if x is connected to a literal v by a chain of supported links, x will also be connected to v by a sequence of basic supported links. This can be shown by first constructing the set of links supported by a derivation and then by removing, one by one, those links $y \cdot z$ which do not comply with the property of definition 2. In particular, we get that:

Lemma 6 Let $D^i_{x,y}$ be a derivation of y from x. Then there is chain of basic links supported by $D^i_{x,y}$ which connects x to y.

We are ready now to prove the consistency theorem:

Theorem (Consistency) If Γ is safe, then Γ is consistent.

Proof Assume otherwise that Γ is safe and that there is a pair of literals x and y, such that $x \mathrel{\not\vdash_\Gamma} y$ and $x \mathrel{\not\vdash_\Gamma} \sim y$. We will assume, furthermore, that there is no derivation $D^j_{x',y'}$ of length shorter than $D^i_{x,y}$ such that a counter-derivation $D^k_{x',\sim y'}$ can be constructed. We prove safeness and inconsistency to be contradictory by induction on the length of $D^i_{x,y}$. If the length of $D^i_{x,y}$ is one, it means that there is a link $x \cdot y$ in Γ. Moreover, since $x \mathrel{\not\vdash_\Gamma} \sim y$, x must defeat the link $x \cdot y$. That is, there must be a link $w \cdot \sim y$ such that either $w = x$ or $w \mathrel{\not\vdash_\Gamma} x$ and $x \mathrel{\not\vdash_\Gamma} w$. In either case, Γ^* must include a cyclic path comprising the antecedents w and x of two conflicting links, contradicting the assumption that Γ is safe.

Assume now instead that the length of $D^i_{x,y}$ is n, $n > 1$, and thus, that no derivation of length smaller than n can be contradicted. Let $D^k_{x,\sim y}$ be a derivation that contradicts $D^i_{x,y}$, and let $\tau = t_1 \cdot t_2 \cdots t_n$, with $t_1 = x$ and $t_n = \sim y$, be a path connecting x to $\sim y$ formed by chaining basic links supported by $D^k_{x,\sim y}$. Since the derivation $D^i_{x,y}$ establishes y from x, it must defeat the conflicting path τ. That is, $D^i_{x,y}$ must support a literal w such that a link $w \cdot \sim t_i$, $1 < i \leq n$, is in Γ, and where $w * t_{i-1}$. On the other hand, $D^k_{x,\sim y}$ supports t_i and, therefore, must defeat all paths connecting x to $\sim t_i$ and, in particular, those paths of the form $x \cdot \sigma \cdot w \cdot \sim t_i$. $D^k_{x,\sim y}$, however, cannot defeat all paths of the form $x \cdot \sigma \cdot w$. Due to the fact that w is supported by $D^i_{x,y}$, there must be a path $x \cdot \sigma' \cdot w$ all of whose σ' literals are also supported by $D^i_{x,y}$. If all paths $x \cdot \sigma \cdot w$ were defeated by x, lemma 5 would imply that either $x \mathrel{\not\vdash_\Gamma} \sim w$ or $x \mathrel{\not\vdash_\Gamma} \sim v$, for a literal v in σ', contradicting the minimality of $D^i_{x,y}$. Thus, there must be a path $x \cdot \sigma'' \cdot w$ which is not defeated by x. However, since $D^k_{x,\sim y}$ supports t_i, it must defeat the extended path $x \cdot \sigma'' \cdot w \cdot \sim t_i$. $D^k_{x,\sim y}$ must, therefore, support a literal t such that $t \cdot t_i$ and $t * w$. Now, from $t * w$, $w * t_{i-1}$ and the fact that the $t_{i-1} \cdot t_i$

is a basic link supported by $D^k_{x,\sim y}$, we obtain that either $t_{i-1} * t$ or t_{i-1} must be equal to t. In either case, we obtain a cyclic path embracing the antecedents of the conflicting links $t_{i-1} \cdot t_i$ and $w \cdot \sim t_i$, contradicting the assumption that Γ is safe. ∎

Theorem (Correspondence) In a safe network Γ, $x \mathrel{\not\vdash}_\Gamma y$ if and only if there is a certified path $x \cdot \sigma \cdot y$ in Γ^*.

Proof Let $x \cdot \sigma \cdot y$ be a chain of links supported by a derivation $D^i_{x,y}$. If all the paths connecting x to y are defeated by x, then, by lemma 5, either $x \mathrel{\not\vdash} \sim y$ or there is a literal v along σ such that $x \mathrel{\not\vdash} \sim v$. In either case, the network is inconsistent and, therefore, by the result above, not safe. ∎

Theorem (Complexity) The problem of determining whether $x \mathrel{\not\vdash}_\Gamma y$ for an arbitrary Γ is NP-Complete.

Proof The theorem is proved by constructing a polynomial reduction from an NP-Complete problem, 3SAT [Garey and Johnson, 79]. The reduction has two parts, in the first part we construct an inheritance net Γ for an arbitrary instance P of the 3SAT problem. In the second part we prove that $x \mathrel{\not\vdash}_{D_\Gamma} y$ holds if and only if P is not satisfiable.

Let $P = (v_{11} \vee v_{12} \vee v_{13}) \wedge (v_{21} \vee v_{22} \vee v_{23}) \wedge \cdots \wedge (v_{n1} \vee v_{n2} \vee v_{n3})$ be an arbitrary instance of the 3SAT problem. P is a conjunction of n clauses, each of which is a disjunction of three literals. P is satisfiable if and only if there is a truth valuation that makes P true.

The core of the network $\Gamma = \Gamma_P$ is formed by n layers L_i, $i = 1, \ldots, n$, of three positive literals or nodes p_{ij}, $j = 1, 2, 3$ which correspond to the literals v_{ij} in the i-th clause of P[9] (fig. refproof). Layer i, for $i = 1, \ldots, n-1$ is fully connected to layer $i+1$ by positive links. Furthermore, we add links $x \cdot p_{1j}$ and $p_{nj} \cdot \sim y$, for $j = 1, 2, 3$. Additionally, for every pair of literals v_{ij} and v_{kl} in P such that $v_{ij} = \sim v_{kl}$ and $i < k$, we allocate in Γ two nodes p'_{ijkl} and p''_{ijkl} as displayed in fig. 6. Finally, a path $x \cdot q \cdot y$ is added to the resulting net (fig. 6).

We will prove the target theorem by showing a series of simple lemmas. We use the fact that Γ is safe and, therefore, consistent. As a result, if $x \mathrel{\not\vdash} w$ follows, we are guaranteed that $x \mathrel{\not\vdash} \sim w$ does not. ∎

Lemma 7 $\mathrel{\not\vdash}_\Gamma \langle p_{ij} | p'_{ijkl} | p_{kl} \rangle$ if and only if $v_{ij} = \sim v_{kl}$ and $i < k$.

Note that the 'if' part trivially follows from the placing of p' nodes in Γ. The 'only if' part follows from the placing of p'' nodes which guarantee that each p' is involved in a single domination triplet between p's.

Lemma 8 $\mathrel{\not\vdash}_\Gamma \langle p_{ij} \| x \| p_{kl} \rangle$ if and only if $v_{ij} = \sim v_{kl}$ and $i < k$.

First of all, notice that all the domination triplets in Γ of the form $\langle p_{ij} | w | p_{kl} \rangle$ have w equal to either a p' node or a p'' node. Furthermore, from the construction of Γ, both

[9] This technique was reported in [Gabow *et al.*, 79] and came to our attention via [Selman, 89].

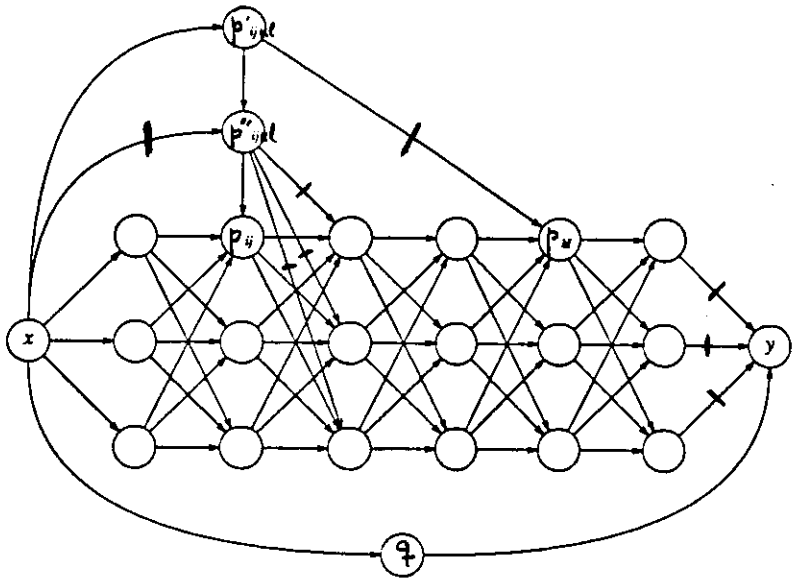

Figure 6: $x \not\vdash_\Gamma y$ if and only if P is not satisfiable

$x \not\vdash_\Gamma p'_{ijkl}$ and $x \not\vdash_\Gamma \sim p''_{ijkl}$ follow, for all p'_{ijkl} and p''_{ijkl} in the net. As a result, we get that $\not\vdash_\Gamma \langle p_{ij} \| x \| p_{kl} \rangle$ if and only $\not\vdash_\Gamma \langle p_{ij} | p'_{ijkl} | p_{kl} \rangle$, which together with lemma 7 lead to the target result.

Definition 2. Let us say that a path is P-consistent if it does not involve p' and p'' literals nor pair of literals p_{ij} and p_{kl} such $v_{ij} = \sim v_{kl}$ in P.

Lemma 9 P is satisfiable if and only if there is a P-consistent path connecting x to $\sim y$.

A path in Γ^* connecting x to $\sim y$ via p_{ij} nodes corresponds to a selection of single literals v_{ij} from each clause C_i $i = 1, \ldots, n$ in P and, vice versa, each such selection corresponds to a path from x to $\sim y$ via p_{ij} nodes. A P-consistent path from x to $\sim y$ therefore exists, if and only if it is possible to select a literal from each clause in P such that no complementary pair of literals is selected. This is in turn equivalent to the existence of a truth assignment to the literals in P such that P is satisfied.

Lemma 10 Γ^* contains a P-consistent path from x to $\sim y$ if and only if it contains a path from x to $\sim y$ undefeated by x.

Let σ be a path in Γ^* connecting x to $\sim y$. σ may or may not involve p' and p'' nodes. In the former case, σ must include a link of the form $p'_{ijkl} \cdot p''_{ijkl}$. It is therefore easy to show that σ is defeated by x as, indeed, by construction, x dominates those links. Thus, if a path is undefeated it must therefore be free from p''s and p'''s. Lemma 10 then results from lemma 8 and the definition of P-consistency.

Since q is connected to x by a direct link, $x \not\vdash_\Gamma y$ can be derived if and only if all paths

connecting x to $\sim y$ are defeated by x. Since the mapping from P to Γ incurs in polynomial time and space, the main theorem thus immediately follows from lemmas 9 and 10. ∎

B Code

```
;;;
;;; BUILDING THE NET
;;;

(defmacro define-net (name paths)
  "Defines a net called <name> with the links
   in <paths>"
  `(let ((net-paths ,paths))
     (setq ,name (make-net :paths net-paths))
     (dolist (path net-paths)
       (assimilate-path path ,name))
     (print-net ,name)))

;;;
;;; Data Structures
;;;

(defstruct net
  paths
  literals)


(defstruct literal
  name
  parents
  sons
  complement
  status
  conseqs)

;;;
;;; Utilities
;;;

(defmacro first-name (path)
  `(car ,path))

(defmacro end-path? (path)
  `(null (cdr ,path)))

(defconstant *neg-sign* #\-)

(defmacro negative-sign? (x)
  `(char= ,x *neg-sign*))

(defmacro positive-sign? (x)
  `(not (negative-sign? ,x)))

(defmacro sign (string-name)
  `(elt  ,string-name 0))

(defmacro pos-name (name string-name)
  `(if (negative-sign? (sign ,string-name))
       (intern (subseq ,string-name 1))
     ,name))

(defmacro marked? (lit mark)
  `(equal ,mark (literal-status ,lit)))

(defmacro mark-paths (source wall)
  `(propagate t ,source ,wall))

(defmacro clean-paths (source wall)
  `(propagate nil ,source ,wall))
```

```
(defun assimilate-path (path net)
  "Adds the link(s) in <path> to <net>"
  (labels
     ((assimilate (parent rest)
        (when rest
          (let ((son
            (get-literal (first-name rest) net)))
              (push son (literal-sons parent))
              (push parent (literal-parents son))
              (assimilate son (cdr rest))))))
   (cond ((end-path? path)
          (error "~% ~S not a valid path ~%" path))
         (t
          (assimilate
           (get-literal (first-name path) net)
           (cdr path))))))

(defun get-literal (name net)
  "Retrieves literal structure given its <name>"
  (multiple-value-bind
    (sign pos-name) (sign-and-name name)
    (let ((pos-lit
           (or
            (find pos-name
                  (net-literals net)
                  :key #'literal-name)
            (create-pos-literal pos-name net))))
     (cond
      ((positive-sign? sign)
       pos-lit)
      ((literal-complement pos-lit))
      (t
       (let ((neg-lit
              (create-neg-literal name net)))
         (setf (literal-complement pos-lit) neg-lit)
         (setf (literal-complement neg-lit) pos-lit)
         neg-lit))))))


(defun sign-and-name (name)
  "Returns polarity of literal + atom"
  (let ((name-string (symbol-name name)))
    (values
     (sign name-string)
     (pos-name name name-string))))


(defun create-pos-literal (name net)
  (let ((literal (make-literal :name name)))
    (push literal (net-literals net))
    literal))

(defun create-neg-literal (name net)
  (make-literal :name name))


(defun print-net (net)
```

```
   (dolist (parent (net-literals net))
     (when (literal-sons parent)
        (let ((parent-name (literal-name parent))
              (son-names
                (mapcar #'literal-name
                        (literal-sons parent))))
          (dolist (son son-names)
            (format t "~A -> ~A ; " parent-name son))
          (terpri)))))


;;;
;;; RUN TIME
;;;


(defun run (net)
  "Computes consequences of each pos-literal
   in <net> (by iterations)"
  (do ((i 1 (1+ i))
       (conseqs)
       (new-conseq? t))
      ((null new-conseq?) 'DONE)
      (format t "~%~%-*-*- Iteration ~D -*-*-~%~%" i)
      (setq new-conseq? nil)
      (dolist (lit (net-literals net))
        (setq conseqs (derive-new-conseqs lit net))
        (setq new-conseq? (or new-conseq? conseqs))
        (print-new-consequences lit conseqs))))


(defun derive-new-conseqs (lit net)
  "Attempts to expand consequences of literal"
  (do* ((parent-boundary
          (cons lit (literal-conseqs lit)))
        (parent
          (car parent-boundary)
          (car parent-boundary))
        (new-consequences))
      ((null parent-boundary)
       new-consequences)
      (pop parent-boundary)
      (dolist (son (literal-sons parent))
        (when (new-conseq? son lit (wall son  lit))
           (push son new-consequences)
           (push son parent-boundary)
           (push son (literal-conseqs lit))))))


(defun new-conseq? (target source wall)
  "Tests whether <target> follows from <source> by
   checking whether markers get to ~<target>
   from source"
  (and
   (not (member target (literal-conseqs source)))
   (or
      (not (literal-complement target))
      (prog2
        (mark-paths source wall)
        (not (reached? (literal-complement target)))
        (clean-paths source wall)))))

(defun reached? (lit)
  (literal-status lit))

(defun wall (target source)
  "Computes the set of literals that can stop markers
   without affecting undefeated paths"
  (do*
   ((poss-parents
      (literal-parents target) (cdr poss-parents))
    (parent (car poss-parents) (car poss-parents))
    (barrier))
   ((null poss-parents)
    (if (literal-complement target)
      (pushnew (literal-complement target) barrier)
       barrier))
    (when
      (or
         (equal parent source)
         (member parent (literal-conseqs source)))
      (pushnew parent barrier)
      (setq
        barrier
        (union barrier (literal-conseqs parent))))))

(defun propagate (mark source wall)
  "Recursively propagates <mark> from <source>
   until hitting <wall>"
  (labels  ((prop-mark (lit)
    (cond ((or (marked? lit mark)
               (member
                 (literal-complement lit)
                 (literal-conseqs source))
               (member lit wall))
           (setf (literal-status lit) mark))
          (t
           (setf (literal-status lit) mark)
           (dolist (son (literal-sons lit))
             (prop-mark son))))))
   (prop-mark source)))



(defun print-new-consequences (lit consequences)
  (when consequences
     (format t "New consequences from ~A:~{ ~A ~} ~%"
        (literal-name lit)
        (mapcar #'literal-name consequences))))
```