

# On the Equivalence of Upward and Downward Inheritance Reasoners

Krishnaprasad Thirunarayan \*

Department of Computer Science and Engineering

Wright State University, Dayton, OH 45435.

Email: tkprasad@cs.wright.edu

## Abstract

In this paper, we analyze systematically, the downward (property flow) and the upward (individual flow) views of inheritance for different categories of inheritance networks. We observe that both these views assign the same meaning to tree-structured hierarchies, and explain the divergence in the interpretation of more general networks in terms of their expressive power. This simple analysis sheds light on the inherent nature of nonmonotonic inheritance and can form the basis for the design of efficient algorithms for certain classes of queries. In addition, we describe the notion of preferential inheritance to specify additional conflict resolution information that can be integrated smoothly with the upward view of inheritance.

## 1 Introduction

Inheritance networks are DAGs with two types of nodes — *individual* nodes and *property* nodes. The arcs between these nodes come in four flavors: A solid arc from an individual node to a property node represents an *instance-of*

---

\*This research was supported in part by the NSF grant IRI-9009587.

relation, while a “slashed” solid arc from an individual node to a property node represents *not-instance-of* relation. A solid arc from a property node to a property node represents an *is-a* relation, while a “slashed” solid arc from a property node to a property node represents *is-not-a* relation. We orient these networks such that the individual nodes are at the bottom and the arcs to property nodes are directed upwards.

There have been a number of proposals for the semantics of inheritance networks — done either through a translation into a general logical formalism [1, 5, 8, 9] or by special path-based techniques [2, 3, 13, 15]. These approaches formalize differing intuitions about inheritance, and consequently, they interpret the same network topology differently. For example, one can visualize the inheritance of a property by an individual in two ways (as described in Section 3). One view is that individuals move up the network “acquiring” properties (also termed as forward chaining or upward flow or bottom-up approach) [2]. Another view is that of properties flowing down the network “imposing” themselves on individuals (also called backward chaining or downward flow or top-down approach) [15]. [16] gives a comparison between these two views of inheritance reasoning; [11] studies their computational complexity. However, as far as we know, there has been no systematic investigation of the relationship between the two views of inheritance as a “function” of the expressive power of the inheritance networks. In this paper we propose to explore this issue, to provide a deeper understanding of the inherent nature of nonmonotonic inheritance.

We motivate the evolution of inheritance networks in Section 2, and describe a simple framework for specifying inheritance in Section 4. The semantics specification we write, relate the meaning of a node to the meaning associated with its neighbors. We then study how well these “local” semantic constraints capture the intuitive notion of inheritance for different categories of networks. We show in Section 4.1 that the two views of inheritance are equivalent for tree-structured hierarchies, and explain their divergence for more expressive networks in Section 5. This equivalence result can form the basis for efficient query processing. For instance, one can use the downward view to compute properties inherited by an individual, and the upward view to

compute individuals possessing a property. Finally, in Section 5.5, we describe preferential networks to specify additional conflict resolution information that can be integrated smoothly with the upward view of inheritance. This fact illustrates certain practical advantages of formalizing the upward view over the downward view.

## 2 The Evolution of Inheritance Networks

Representing knowledge about the world by enumerating explicitly the individuals and the properties they possess, lacks *notational efficacy*. So, we define suitable abstractions based on the commonalities among individuals at different levels of detail, and organize them as *tree-structured class hierarchies*. Here, an individual *inherits* a property from its parent class. These hierarchies can be further generalized to represent *redundant arcs*, to permit explicit representation of facts that are implicitly supported by the network. But the tree-structure leads to needless duplication (resulting in storage inefficiencies and update problems) when used to represent abstractions that multiply inherit from orthogonal abstractions. *Tangled hierarchies* were introduced to permit multiple parents [15].

Any realistic representation of real-world knowledge must necessarily allow for *exceptions*, which give rise to contradictory inheritance [15]. In tree-structured class hierarchies the contradiction can be resolved using specificity information. But, incorporation of multiple inheritance and exceptions in tangled hierarchies introduces *ambiguity*. Furthermore, if we permit chains of defeasible conclusions, then we obtain the *traditional inheritance networks*. Here, the topology of the network can be ambiguous even when there is no inherent ambiguity in the knowledge being represented. This motivates *preferential networks* that permit specification of additional disambiguation information [8].

### 3 Property Flow View vs Individual Flow View

We now describe the downward and the upward views of inheritance informally [2, 16]. A word about the notation used in Figures 1 and 2. The solid lines stand for positive arcs and the slashed lines stand for negative arcs. The letter  $i$  stands for an individual, while the letters  $p$ ,  $q$ , and  $r$  represent properties. In the upward view,  $i \in p^+$  (resp.  $i \in p^-$ ) says that “individual  $i$  acquires property  $p$  (resp.  $\neg p$ )”; in the downward view,  $r \in p^+$  (resp.  $r \in p^-$ ) says that “property  $r$  (resp.  $\neg r$ ) is imposed on  $p$ ”.

Figure 1(a) depicts how individuals and properties propagate through a (positive) arc from node  $q$  to node  $p$  in an exception-free network. Because every  $q$  is a  $p$ , all the individuals that acquire property  $q$  also acquire property  $p$ . Similarly, all the properties that are imposed on  $p$ 's are also imposed on  $q$ 's. These two views can be considered as *duals* of each other.

In the presence of exceptions however, the flow of individuals and properties through positive and negative arcs is not symmetric. Consider the case of a positive arc from  $q$  to  $p$ . In the upward view, all the individuals that acquire  $q$  also acquire  $p$  but none of the individuals that acquire  $\neg q$  propagate through this arc. This is because  $q$  is a  $p$  does not say anything about the association of property  $p$  with individuals that do not possess property  $q$ . In the downward view, all the properties that are imposed on  $p$  are imposed on  $q$ . These cases are depicted in Figures 1(a) and 1(b). Now, let the arc from  $q$  to  $p$  be negative. In the upward view, all the individuals that acquire  $q$  also acquire  $\neg p$  but none of the individuals that acquire  $\neg q$  pass through this arc. In the downward view, all the properties imposed on  $p$  are blocked through this arc, that is, they do

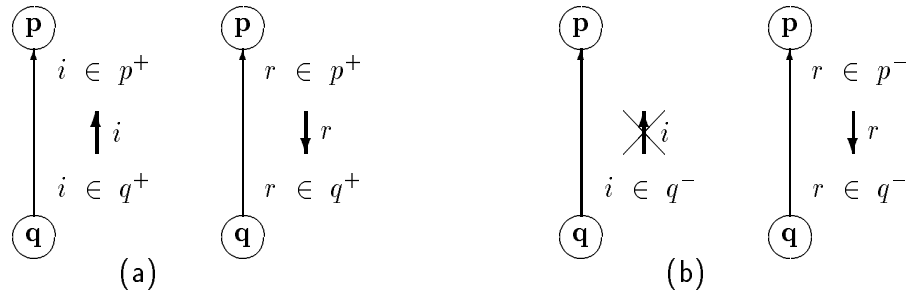


Figure 1: Flow through positive arcs

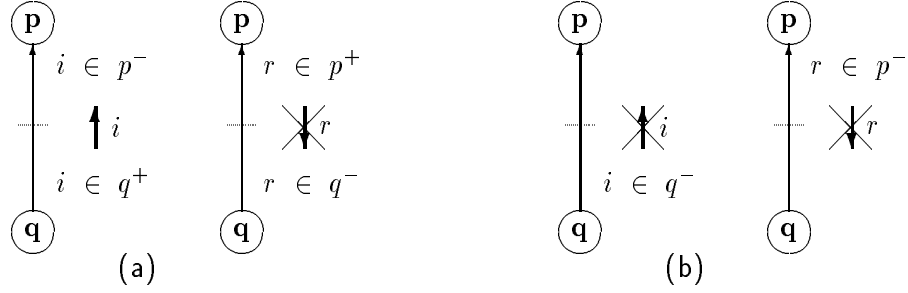


Figure 2: Flow through negative arcs

not propagate to  $q$ , but  $\neg p$  is imposed on  $q$ . These cases are shown in Figures 2(a) and 2(b).

## 4 Direct Semantics

One can give a direct set-based semantics to networks by defining what a *semantic structure* for a network is, and by specifying what it means for a structure to be a model [8]. The definition of a semantic structure depends on the view of inheritance. In order to formalize the upward view of inheritance, one can associate sets of individuals with each class/property node. In contrast, to formalize the downward view, we associate sets of properties with each individual/class node.

### 4.1 Tree-Structured Class Hierarchies

A *tree-structured class hierarchy* consists of a finite set of individual nodes  $\mathbf{I}$ , a finite set of class nodes  $\mathbf{C}$ , a finite set of property nodes  $\mathbf{P}$ , a parent function  $\mathbf{par}: (\mathbf{I} \cup \mathbf{C}) \mapsto \mathbf{C}$ , a set of positive arcs  $\mathbf{E}^+ \subseteq (\mathbf{I} \cup \mathbf{C}) \times \mathbf{P}$  and a set of negative arcs  $\mathbf{E}^- \subseteq (\mathbf{I} \cup \mathbf{C}) \times \mathbf{P}$ . (Here, the nodes  $\mathbf{I} \cup \mathbf{C}$  together with the arcs given by the  $\mathbf{par}$ -function, form an inverted tree.)

A word about the notation used below. We write  $E^+(i, p)$  (resp.  $E^-(i, p)$ ) for  $(i, p) \in E^+$  (resp.  $(i, p) \in E^-$ ), and  $\neg E^+(i, p)$  (resp.  $\neg E^-(i, p)$ ) for  $(i, p) \notin E^+$  (resp.  $(i, p) \notin E^-$ ).

### 4.1.1 Downward Semantics

The *downward structure*  $\mathcal{M}$  for a tree-structured hierarchy has two components — a class membership function  $\mathcal{M}_m$  and a property inheritance function  $\mathcal{M}_i$ .  $\mathcal{M}_m$  associates with each individual  $i \in \mathbf{I}$ , a subset  $i_m$  of  $\mathbf{C}$  representing the classes to which the individual belongs.  $\mathcal{M}_i$  associates with each individual  $i \in \mathbf{I}$ , a pair of subsets of  $\mathbf{P}$ , denoted as  $(i^+, i^-)$ , where the set  $i^+$  (resp.  $i^-$ ) contains properties (resp. not) possessed by  $i$ .  $\mathcal{M}_m$  associates with each class  $c \in \mathbf{C}$ , a subset  $c_m$  of  $\mathbf{C}$  representing the superclasses of  $c$ .  $\mathcal{M}_i$  associates with each class  $c \in \mathbf{C}$ , a pair of subsets of  $\mathbf{P}$ , denoted as  $(c^+, c^-)$ , where the set  $c^+$  (resp.  $c^-$ ) contains properties (resp. not) possessed by typical members of  $c$ . The constraints that a downward structure must satisfy in order to be a model are as follows.

- Individual  $i$  belongs to class  $c$  if  $c$  is the parent of  $i$ , or if  $i$ 's parent class is a subclass of  $c$ . Formally:  $c \in i_m$  **if**  $c = \mathbf{par}(i) \vee c \in \mathbf{par}(i)_m$ .  
Similarly for a class  $d$  to be a subclass of class  $c$ .

- Individual  $i$  possesses property  $p$  (resp.  $\neg p$ ) if there is a positive (resp. negative) arc from  $i$  to  $p$ , or if  $i$ 's parent class possesses property  $p$  (resp.  $\neg p$ ) and there is no negative (resp. positive) arc from  $i$  to  $p$ . The latter condition allows an explicitly stated property to override an inherited property in the event of a conflict. Formally:

$$p \in i^+ \text{ **if** } \mathbf{E}^+(i, p) \vee [p \in \mathbf{par}(i)^+ \wedge \neg \mathbf{E}^-(i, p)].$$

$$p \in i^- \text{ **if** } \mathbf{E}^-(i, p) \vee [p \in \mathbf{par}(i)^- \wedge \neg \mathbf{E}^+(i, p)].$$

Similarly for a (typical member of) class  $c$  to possess property  $p$  or  $\neg p$ .

The minimality constraints are imposed by replacing **if** with **iff**, which associates with each tree-structured hierarchy a minimum model as its meaning, as shown in Theorems 1 and 2.

### 4.1.2 Upward Semantics

An *upward structure*  $\mathcal{M}$  for a tree-structured hierarchy has two components — a membership function  $\mathcal{M}_m$  and a subclass function  $\mathcal{M}_s$ .  $\mathcal{M}_m$  associates with

each class  $c \in \mathbf{C}$ , a subset  $c_m$  of  $\mathbf{I}$  representing the individuals that belong to class  $c$ .  $\mathcal{M}_m$  associates with each property  $p \in \mathbf{P}$ , a pair of subsets  $(p_m^+, p_m^-)$  of  $\mathbf{I}$ , where  $p_m^+$  (resp.  $p_m^-$ ) represents the set of individuals that possess property  $p$  (resp.  $\neg p$ ).  $\mathcal{M}_s$  associates with each class  $c \in \mathbf{C}$ , a subset of  $\mathbf{C}$ , denoted as  $c_s$ , where the set  $c_s$  contains subclasses of class  $c$ .  $\mathcal{M}_s$  associates with each property  $p \in \mathbf{P}$ , a pair of subsets of  $\mathbf{C}$ , denoted as  $(p_s^+, p_s^-)$ , where the set  $p_s^+$  (resp.  $p_s^-$ ) contains classes whose “typical” members possess  $p$  (resp.  $\neg p$ ). The constraints that an upward structure must satisfy to be a model are as follows.

- Individual  $i$  belongs to class  $c$  if  $i$  is a child of  $c$ , or  $i$  belongs to a child of  $c$ . Formally:  $i \in c_m$  **if**  $c = \mathbf{par}(i) \vee [\exists e : c = \mathbf{par}(e) \wedge i \in e_m]$ .

Similarly for a class  $d$  to be a subclass of a class  $c$ .

- Individual  $i$  possesses property  $p$  (resp.  $\neg p$ ) if there is a positive (resp. negative) arc from  $i$  to  $p$ , **or** if there is no negative (resp. positive) arc from  $i$  to  $p$  **and** there is a class  $e$  that has a positive (resp. negative) arc to  $p$  and to which  $i$  belongs, such that  $e$  is more specific than all the classes that have a negative (resp. positive) arc to  $p$  and that contain  $i$ .

$$i \in p_m^+ \text{ **if** } \mathbf{E}^+(i, p) \vee \{ \neg \mathbf{E}^-(i, p) \wedge [\exists e : \mathbf{E}^+(e, p) \wedge i \in e_m \wedge [\forall d : e \neq d \wedge \mathbf{E}^-(d, p) \wedge i \in d_m \Rightarrow e \in d_s]] \}.$$

$$i \in p_m^- \text{ **if** } \mathbf{E}^-(i, p) \vee \{ \neg \mathbf{E}^+(i, p) \wedge [\exists e : \mathbf{E}^-(e, p) \wedge i \in e_m \wedge [\forall d : e \neq d \wedge \mathbf{E}^+(d, p) \wedge i \in d_m \Rightarrow e \in d_s]] \}.$$

Similarly for a class  $c$  to possess property  $p$  (resp.  $\neg p$ ).

The minimality constraints are imposed by replacing **if** with **iff**, which associates with each network a minimum model as its meaning, as shown in Theorems 3 and 4.

We now show that the upward and the downward meaning ascribed to tree-structured hierarchies support the same set of conclusions about properties inherited by individuals, and the same generic statements about properties associated with classes.

**Definition 1** *There is an s-path from an individual  $i$  (resp. a class  $c$ ) to a class  $d$  if there exists a  $k \geq 1$  such that  $\mathbf{par}^k(i) = d$  (resp.  $\mathbf{par}^k(c) = d$ ). (The length of the s-path from  $c$  to  $d$  is denoted as  $\text{len}(c,d)$ .)*

**Definition 2** *The depth of an individual/class  $c$  is the length of the longest s-path originating at  $c$ . That is,*

$$\text{depth}(c) = \begin{cases} 0 & \text{if } \mathbf{par}(c) \text{ is undefined} \\ \text{depth}(\mathbf{par}(c)) + 1 & \text{otherwise} \end{cases}$$

**Definition 3** *The height of an individual (or a class with no in-arcs) is 0. The height of a class  $c$  (with in-arcs) is the length of the longest s-path from an individual (or a class with no in-arcs) to  $c$ . That is,  $\text{height}(c) = 1 + \max(\{\text{height}(d) \mid \mathbf{par}(d) = c\})$ .*

**Theorem 1** *The minimality constraints given in Section 4.1.1 associate a unique minimal model with tree-structured hierarchies.*

**Proof:** The proof is by induction on the depth of a node.

**Basis:** If the *depth* of a class/individual node  $c$  is 0, then  $p \in c^+$  (resp.  $p \in c^-$ ) holds iff there is a positive (resp. negative) arc from  $c$  to  $p$  (resp.  $p$ ).  $c_m = \emptyset$ .

**Induction Step:** If  $\text{depth}(c) = k + 1 (> 0)$ , then the meaning of  $c$  is completely and uniquely determined by the out-arcs and the meaning of its parent, as evident from the constraint. In particular,  $p \in c^+$  (resp.  $p \in c^-$ ,  $d \in c_m$ ) holds iff there is a positive (resp. negative, positive) arc from  $c$  to  $p$  (resp.  $p$ ,  $d$ ), or if its parent possesses  $p$  (resp.  $\neg p$ ,  $d$ ) that has not been overridden by a direct arc. •

**Theorem 2** *If the set of positive arcs  $\mathbf{E}^+$  and the set of negative arcs  $\mathbf{E}^-$  are disjoint, then so are the pairs of sets  $(c^+, c^-)$  and  $(i^+, i^-)$ .*

**Proof:** The proof is by induction on the depth of a class/individual node.

**Basis:** If the *depth* of a class/individual node  $c$  is 0, then  $(c^+, c^-)$  are disjoint given  $(E^+, E^-)$  are disjoint.

**Induction Step:** Let  $\text{depth}(c) = k + 1 (> 0)$ . If  $(c^+, c^-)$  are not disjoint, then it must be the case that  $(\mathbf{par}(c)^+, \mathbf{par}(c)^-)$  are not disjoint. This is because we are given that  $(\mathbf{E}^+, \mathbf{E}^-)$  are disjoint, and it is not possible for  $\mathbf{E}^+(c, p)$  (resp.

$\mathbf{E}^-(c, p)$ ) and  $p \in \mathbf{par}(c)^- \wedge \neg \mathbf{E}^+(c, p)$  (resp.  $p \in \mathbf{par}(c)^+ \wedge \neg \mathbf{E}^-(c, p)$ ) to hold simultaneously, because a direct arc always overrides any conflicting inheritance through the parent. The implication that  $(\mathbf{par}(c)^+, \mathbf{par}(c)^-)$  intersect, contradicts the induction hypothesis. So,  $(c^+, c^-)$  are disjoint. •

**Theorem 3** *The minimality constraints given in Section 4.1.2 associate a unique minimal model with tree-structured hierarchies.*

**Proof:** A proof by induction on the height of a class node along the lines of Theorem 1 shows that the meaning associated with the class nodes and the property nodes are unique. •

**Theorem 4** *If the set of positive arcs  $\mathbf{E}^+$  and the set of negative arcs  $\mathbf{E}^-$  are disjoint, then so are the pairs of sets  $(p_m^+, p_m^-)$  and  $(p_s^+, p_s^-)$ .*

**Proof:** The class hierarchy has an inverted tree structure, so the ancestors of an individual/class can be *totally ordered* by the reachability relationship. Thus it will never be the case that the second disjunct of the constraints for  $i \in p_m^+$  and  $i \in p_m^-$  (resp.  $c \in p_s^+$  and  $c \in p_s^-$ ) are true simultaneously. Furthermore, it is not possible for both the first disjunct of one and the second disjunct of the other to hold at the same time, because a direct arc always overrides a conflicting inheritance via an ancestor. Thus, if  $\mathbf{E}^+$  and  $\mathbf{E}^-$  are disjoint, then so are  $(p_m^+, p_m^-)$  and  $(p_s^+, p_s^-)$ . •

**Lemma 1**  *$d \in c_m$  (resp.  $d \in i_m$ ) is in the downward meaning iff there exists an s-path from  $c$  (resp.  $i$ ) to  $d$ .*

**Proof:** ( $\Leftarrow$ ) The proof is by induction on the *path length*.

( $\Rightarrow$ ) The proof is by induction on the *depth* of a class. •

**Lemma 2**  *$c \in d_s$  (resp.  $i \in d_m$ ) is in the upward meaning iff there exists an s-path from  $c$  (resp.  $i$ ) to  $d$ .*

**Proof:** ( $\Leftarrow$ ) The proof is by induction on the *path length*.

( $\Rightarrow$ ) The proof is by induction on the *height* of a class. •

**Theorem 5**  *$i \in c_m$  (resp.  $c \in d_s$ ) is in the upward meaning iff  $c \in i_m$  (resp.  $d \in c_m$ ) is in the downward meaning.*

**Proof:** This follows from Lemmas 1 and 2. •

**Definition 4** *There is a positive (resp. negative) d-path from an individual/class  $c$  to a property  $p$  if  $\mathbf{E}^+(c, p)$  (resp.  $\mathbf{E}^-(c, p)$ ) holds, or there exists an s-path from  $c$  to  $d$  and  $\mathbf{E}^+(d, p)$  (resp.  $\mathbf{E}^-(d, p)$ ) holds.*

**Definition 5** *The depth of a class (resp. an individual)  $c$  with respect to a property  $p$  is the length of the shortest d-path from  $c$  to  $p$ , if such a path exists. That is,*

$$\text{depth}_p(c) = \begin{cases} 1 & \text{if } \mathbf{E}^-(c, p) \vee \mathbf{E}^+(c, p) \\ \text{depth}_p(\mathbf{par}(c)) + 1 & \text{otherwise} \end{cases}$$

**Lemma 3**  $p \in c^+$  (resp.  $p \in c^-$ ,  $p \in i^+$ ,  $p \in i^-$ ) *is in the downward meaning iff there exists a d-path from  $c$  (resp.  $c, i, i$ ) to  $p$ , and the shortest such path is positive (resp. negative, positive, negative).*

**Proof:** ( $\Leftarrow$ ) The proof is by induction on the length of the shortest *d-path* from  $c$  to  $p$ .

( $\Rightarrow$ ) The proof is by induction on the *depth* of a class with respect to property  $p$ , as shown below. •

**Lemma 4**  $c \in p_s^+$  (resp.  $c \in p_s^-$ ,  $i \in p_m^+$ ,  $i \in p_m^-$ ) *is in the upward meaning iff there exists a d-path from  $c$  (resp.  $c, i, i$ ) to  $p$ , and the shortest such path is positive (resp. negative, positive, negative).*

**Proof:** Recall that,  $c \in p_s^+$  **iff**  $\mathbf{E}^+(c, p) \vee \{ \neg \mathbf{E}^-(c, p) \wedge$

$$[\exists e : \mathbf{E}^+(e, p) \wedge c \in e_s \wedge [\forall d : e \neq d \wedge \mathbf{E}^-(d, p) \wedge c \in d_s \Rightarrow e \in d_s]] \}.$$

( $\Leftarrow$ ) The proof is by induction on the length of the shortest *d-path* from  $c$  to  $p$ , denoted  $\text{len}(c, p)$ , that is positive.

**Basis:** If  $\text{len}(c, p) = 1$ , then  $\mathbf{E}^+(c, p)$  is true. Thus, the above constraint implies that  $c \in p_s^+$  holds.

**Induction Step:** We show that whenever  $\text{len}(c, p) = k+1 (>1)$ ,  $c \in p_s^+$  holds.

Given a shortest *positive d-path* from  $c$  to  $p$  of length  $k+1 (>1)$ ,  $\neg \mathbf{E}^-(c, p)$  and there exists a unique  $e$ , such that  $e = \mathbf{par}(c)$  and  $\text{len}(e, p) = k$ . This follows from the fact that the classes form a directed tree. So, by induction hypothesis,  $e \in p_s^+$  holds. Furthermore, from the constraint we know that one of the following two conditions must be true.

- $\mathbf{E}^+(e, p)$ . In this case  $e$  satisfies our requirements on the most specific class yielding  $c \in p_s^+$ .
- There exists a class  $f$ , which is a superclass of  $d$  (and hence a superclass of  $c$ ), such that  $\mathbf{E}^+(f, p)$ , and  $f$  is the more specific than the superclasses  $d$  of  $c$  that have a negative arc to  $p$ . This again yields  $c \in p_s^+$ .

( $\Rightarrow$ ) The proof is by induction on the *depth* of a class with respect to  $p$ .

**Basis:** For all classes  $c$  such that  $\text{depth}_p(c) = 1$ , if  $c \in p_s^+$  holds, then  $\mathbf{E}^+(c, p)$  is true, and so there exists a shortest *positive d-path* from  $c$  to  $p$ .

**Induction Step:** Whenever  $\text{depth}_p(c) = k+1$  ( $> 1$ ) and  $c \in p_s^+$  holds, it implies that there exists a most specific superclass  $e$  of  $c$ , such that  $\mathbf{E}^+(e, p)$ . This follows from the above constraint and the fact that the tree-structure totally orders all superclasses. The *d-path* from  $c$  to  $p$  via  $e$  is the shortest because if there were to exist an arc to  $p$  from a subclass of  $e$  that is a superclass of  $c$ , then  $\text{depth}_p(c) \leq k$ , which is contrary to our assumption. Thus, there exists a shortest *d-path* from  $c$  to  $p$  which is positive. •

**Theorem 6**  $p \in i^+$  (resp.  $p \in i^-$ ,  $p \in c^+$ ,  $i \in c^-$ ) is in the downward meaning iff  $i \in p_m^+$  (resp.  $i \in p_m^-$ ,  $c \in p_s^+$ ,  $c \in p_s^-$ ) is in the upward meaning.

**Proof:** This follows from Lemmas 3 and 4. •

**Theorem 7** *The downward meaning and the upward meaning of the tree-structured hierarchies are identical.*

**Proof:** This follows from Theorems 5 and 6. •

Similarly to the path-based approaches in [2, 15], the specifications given above do not support contraposition of defeasible arcs. In contrast, the theories in [1, 7] do support contrapositive reasoning. For instance, consider the following facts: *Birds fly. Penguins do not fly. Penguins are birds. Tweety is a penguin. Robin is a bird.* Almost all approaches in the literature conclude that *Robin flies* and *Tweety does not fly*. However the theories in [1, 7] support additional conclusions like *Robin is not a penguin* through contrapositive reasoning from the implicit fact *Robin flies* and the arc *Penguins do not fly*.

## 5 More on Property Flow vs Individual Flow

We now study the relationship between the upward and the downward views of inheritance for different categories of inheritance networks. Because the two views diverge significantly, we will not formally specify all of them in detail. Instead, we explain the divergence through examples.

### 5.1 Exception-Free Networks

In exception-free networks, there are no inheritance conflicts. So, an individual or a class inherits a property whenever there is a directed path from the former node to the latter node in both the interpretations of inheritance.

**Theorem 8** *The upward meaning and the downward meaning associated with exception-free networks are identical.*

### 5.2 Tree-Structured Hierarchies with Redundant Arcs

Even though the discussion in Section 3 provided valuable insights into the nature of inheritance as flow of individuals and properties through the arcs, it still does not give us a satisfactory account of *multiple inheritance*. Consider the following example: *Birds fly. Penguins do not fly. Penguins are birds. Tweety is a penguin. Tweety is a bird.* Here *Tweety* has two parents — *penguin* and *bird*, and there is an inheritance conflict with respect to property *fly*. Observe also that the arc from *Tweety* to *bird* is *redundant* because there is another path from *Tweety* to *bird*.

According to the downward view, Tweety can fly because it is a bird and cannot fly because it is a penguin. To resolve this conflict in favor of the subclass penguins, one can assign a higher weight to the imposition of property fly through the arc adjacent to *penguin* over that through the arc leading to *bird*. In order to compute the meaning of a node one must first determine all specificity relationships among its parents and consider all the arcs to *its parents, simultaneously*. The meaning of the node can be computed using only the meanings of its parents and the specificity relationship among its parents.

The upward view of inheritance also draws conflicting conclusions about Tweety's flying ability and, once again, the contradiction can be resolved in favor of penguins. However, in order to compute individuals that inherit a property, one must look at a property node and all *its children, simultaneously*. The meaning of the node can be computed using only the meanings of its children and the specificity relationship among its children.

In spite of these differences, the semantics of inheritance captured by both the views are identical as shown below.

### 5.3 Tangled Hierarchies

A *tangled hierarchy* consists of a finite set of individual nodes  $\mathbf{I}$ , a finite set of class nodes  $\mathbf{C}$ , a finite set of property nodes  $\mathbf{P}$ , a set of membership/subclass arcs  $\mathbf{S} \subseteq (\mathbf{I} \cup \mathbf{C}) \times \mathbf{C}$ , a set of positive property arcs  $\mathbf{E}^+ \subseteq (\mathbf{I} \cup \mathbf{C}) \times \mathbf{P}$  and a set of negative property arcs  $\mathbf{E}^- \subseteq (\mathbf{I} \cup \mathbf{C}) \times \mathbf{P}$ . (We write  $\mathbf{S}(i, c)$  for  $(i, c) \in \mathbf{S}$ .)

If the arcs in  $\mathbf{S}$  are such that  $\forall i \in \mathbf{I} \cup \mathbf{C} : [\mathbf{S}(i, c) \wedge \mathbf{S}(i, d)] \Rightarrow [c = d]$ , then we obtain a *tree-structured class hierarchy*.

If the arcs in  $\mathbf{S}$  are such that  $\forall i \in \mathbf{I} \cup \mathbf{C} : [\mathbf{S}(i, c) \wedge \mathbf{S}(i, d)] \Rightarrow$  *there is a directed path between  $c$  and  $d$* , then we obtain a *tree-structured class hierarchy with redundant arcs*.

#### 5.3.1 Downward Semantics

A downward structure can be defined analogously as in Section 4.1.1. The constraints that a downward structure must satisfy in order to be a model are:

For individual/class  $i$  and class  $c$ :

$$c \in i_m \text{ if } \mathbf{S}(i, c) \vee [\exists d : \mathbf{S}(i, d) \wedge c \in d_m].$$

For individual/class  $i$  and property  $p$ :

$$p \in i^+ \text{ if } \mathbf{E}^+(i, p) \vee \{ \neg \mathbf{E}^-(i, p) \wedge [\exists c : \mathbf{S}(i, c) \wedge p \in c^+ \wedge [\neg \exists d : c \neq d \wedge \mathbf{S}(i, d) \wedge p \in d^- \wedge c \in d_m]] \}.$$

$$p \in i^- \text{ if } \mathbf{E}^-(i, p) \vee \{ \neg \mathbf{E}^+(i, p) \wedge [\exists c : \mathbf{S}(i, c) \wedge p \in c^- \wedge [\neg \exists d : c \neq d \wedge \mathbf{S}(i, d) \wedge p \in d^+ \wedge c \in d_m]] \}.$$

The minimality constraints are imposed by replacing **if** with **iff**.

### 5.3.2 Upward Semantics

An upward structure can be defined analogously as in Section 4.1.2. The constraints that an upward structure must satisfy in order to be a model are:

For individual  $i$  and class  $c$ :  $i \in c_m$  **if**  $\mathbf{S}(i, c) \vee [\exists d : \mathbf{S}(d, c) \wedge i \in d_m]$ .

For classes  $c$  and  $d$ :  $c \in d_s$  **if**  $\mathbf{S}(c, d) \vee [\exists e : \mathbf{S}(e, d) \wedge c \in e_s]$ .

For individual/class  $i$  and property  $p$ :

$i \in p_m^+$  **if**  $\mathbf{E}^+(i, p) \vee \{ \neg \mathbf{E}^-(i, p) \wedge [\exists c : \mathbf{E}^+(c, p) \wedge i \in c_m \wedge [\neg \exists d : c \neq d \wedge \mathbf{E}^-(d, p) \wedge i \in d_m \wedge d \in c_s]] \}$ .

$i \in p_m^-$  **if**  $\mathbf{E}^-(i, p) \vee \{ \neg \mathbf{E}^+(i, p) \wedge [\exists c : \mathbf{E}^-(c, p) \wedge i \in c_m \wedge [\neg \exists d : c \neq d \wedge \mathbf{E}^+(d, p) \wedge i \in d_m \wedge d \in c_s]] \}$ .

The minimality constraints are imposed by replacing **if** with **iff**.

**Theorem 9** *The constraints specifying the upward and the downward meanings of tree-structured class hierarchies in Sections 5.3.1 and 5.3.2 are identical to the ones given in Sections 4.1.1 and 4.1.2 respectively.*

**Proof:** The result follows trivially because, for tree-structured class hierarchies,  $[\mathbf{par}(i) = c \text{ iff } \mathbf{S}(i, c)]$  and  $[\mathbf{S}(i, c) \wedge \mathbf{S}(i, d) \Rightarrow c = d]$ . •

We now prove the equivalence results for tree-structured class hierarchies with redundant arcs by reducing them to equivalent tree-structured class hierarchies, and then falling back on the results proved in Section 4.1.

An arc  $\mathbf{S}(i, d)$  of an hierarchy  $\Gamma$  is *redundant*, where  $i \in \mathbf{IUC}$ , if there exist in  $\Gamma$  an arc  $\mathbf{S}(i, c)$  and a directed path from  $c$  to  $d$ .

**Lemma 5** *Let  $\Gamma$  be a tree-structured class hierarchy with redundant arcs. Then, the hierarchy  $\Gamma'$ , obtained by deleting all the redundant arcs in  $\Gamma$ , is tree-structured, and is semantically equivalent to  $\Gamma$ .*

**Proof:** First of all, consider the networks  $\Gamma$  and  $\Gamma'$  without the property nodes. All the arcs in both these networks are positive. So, the set of conclusions about class-memberships and class-subclass relationships supported by the two modified networks are identical by Theorem 8.

Now consider property inheritance. In both the downward and the upward semantics, the conflicting conclusions via the redundant arcs are always defeated by the conclusions sanctioned via the most specific class. Thus, both  $\Gamma$  and  $\Gamma'$  support identical conclusions about property inheritance. •

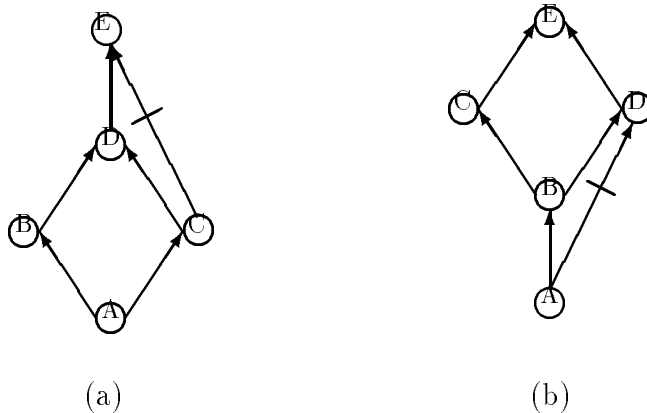


Figure 3: Divergence of upward and downward views

**Theorem 10** *The upward meaning and the downward meaning assigned to tree-structured hierarchies with redundant arcs are identical.*

**Proof:** The equivalence of the two views follows from Lemma 5 and Theorems 7 and 9. •

**Theorem 11** *The minimality constraints of Section 5.3.1 associate a unique minimal model with each tangled hierarchy.*

**Proof:** The proof is by induction on the *depth* of a class/individual. •

**Theorem 12** *The minimality constraints of Section 5.3.2 associate a unique minimal model with each tangled hierarchy.*

**Proof:** The proof is by induction on the *height* of a class/property. •

The *skeptical* theories given here assign a unique meaning to the ambiguous Nixon diamond network: *Quakers are pacifists. Republicans are not pacifists. Nixon is a Quaker. Nixon is a Republican.* In particular, the ambiguity about Nixon’s pacifism is represented explicitly. This is in contrast with the skeptical approach of [2] where the reasoner abstains from drawing any conclusions about Nixon’s pacifism. Note that in our approach the effects of an apparent contradiction are localized.

The upward and the downward skeptical meanings assigned to tangled hierarchies are not identical. For example, in Figure 3(a) from [16], the downward view described above (as well as in [15]) is ambiguous about whether or not

*A's are E's*, while the upward view described above (as well as in [2, 8, 13]) lends support to *A's are not E's*.

Furthermore, it is possible to modify the skeptical constraints described above such that whenever there are conflicting evidences supporting both  $p \in i^+$  or  $p \in i^-$ , exactly one of them is chosen arbitrarily [7]. This modification associates with each network a set of minimal models characterizing the *credulous* meaning. However, the two *credulous* views of inheritance diverge in their interpretation of ambiguous networks. Consider a variant of the Nixon diamond in which *Nixon* represents a class, and the individuals *Dick* and *Pat* are *Nixon*. The upward meaning (as well as in [3, 8]) associates four minimal models with the network stemming from four possible interpretations of the predicate *pacifist* on the individuals *Dick* and *Pat*. In contrast, the downward meaning (as well as in [15]) associates with the network only two minimal models — one in which both *Dick* and *Pat* are pacifists, and another in which they are not. ([16] refers to this phenomenon as *coupling in downward reasoners*.)

## 5.4 Traditional Inheritance Networks

The duality discussed earlier breaks down further when more general networks are considered. In particular, permitting negative arcs into nonterminal nodes makes capturing the upward view of inheritance relatively simpler than the downward view as illustrated by the Wings example [7] below: *Birds fly. Birds are feathered. Flying objects are winged. Tweety is a bird. Tweety does not fly.* The upward meaning supports the fact that Tweety is a bird, has feathers and cannot fly, but it is not known whether Tweety is winged. In contrast, the downward meaning assigns the sets  $(\{winged, fly\}, \emptyset)$  to *fly*, the sets  $(\{fly, winged, feathered, bird\}, \emptyset)$  to *bird* and the sets  $(\{bird, feathered\}, \{fly\})$  to *Tweety*. Intuitively, one can conclude that Tweety is feathered because it is a bird, but not that it is winged. Birds are winged because they fly. But Tweety does not fly, hence it should not inherit wings from bird. In other words, one cannot blindly impose the property *winged* on *Tweety* through *bird*, even though there is no apparent conflict. *The meaning of the node Tweety cannot be determined by looking only at the meaning of its parents.*

*Instead, we need to know the context (in the form of overlapping inheritance paths [15]) to see whether a property (such as wings, in this example) is inherited.* What this implies is that, while it may be possible to specify the upward view of inheritance using “local” constraints, this is not the case when formalizing the downward view.

In addition, the downward view can give counter-intuitive interpretation to certain network topologies [10, 16]. Consider the following example from [16]: *C’s and D’s are E’s. B’s are C’s and D’s. A’s are B’s and not D’s.* See Figure 3(b). In the downward view, there is an ambiguity about whether or not *A’s are E’s*; while the upward view accords support to the conclusion — *A’s are E’s*, which is intuitively satisfactory. The reason for this discrepancy can be traced back to a lack of locality property of the downward meaning.

## 5.5 Preferential Networks

Mostly, the ambiguity in a network is due to incompleteness in our knowledge. This may get resolved as the knowledge evolves. Preferential networks allow one to specify the meta-knowledge about how an ambiguity with respect to a property can be resolved for each pair of children of the property node. This additional information cannot be incorporated naturally in the downward view. On the other hand, in the upward view, one may think of the specificity relationship among the in-arcs into a property node  $p$  as consisting of two components: the inferred specificity constraints implicit in the topology of the network, and the input preferential constraints explicitly input by the representer. Thus, the upward view seems to have “more structure” to it than the downward view.

In order to better understand the notion of preferential inheritance we compare it with the conflict resolution strategies used by object-oriented languages such as Smalltalk, C++, Eiffel, Flavors, Loops, CommonLoops etc.

Firstly, we observe the following correspondence between object hierarchies and inheritance networks: The constants, instance variables and methods associated with a class correspond to property nodes of a network; the members of a class correspond to individual nodes of a network. In case an object-oriented system supports only simple inheritance, the object hierarchies that can arise

correspond to class-property hierarchies; if it supports multiple inheritance, the object hierarchies correspond to tangled hierarchies.

Normally, the methods applicable to an object are all the methods associated with its ancestors (including itself). However, in the presence of exceptions, a method associated with a class may be redefined in a subclass. This results in an inheritance conflict that can be resolved in favor of the method associated with the subclass. This is in accord with the specificity relation defined for tree-structured class hierarchies.

In inheritance networks that support both multiple inheritance and exceptions, there is no one satisfactory ambiguity resolution technique derivable from the network topology. In fact, different object-oriented languages advocate significantly different strategies for disambiguation. In Flavors and Loops, a fixed conflict resolution strategy is built into the system. The class precedence relation in Flavors (resp. Loops) is obtained by doing a depth first traversal of the hierarchy, removing all but the first (resp. last) occurrence of duplicate classes [12]. On the contrary, CommonLoops, C++, and Smalltalk expect the users to provide sufficient disambiguation information, arguing that there is no universally acceptable scheme for automatic conflict-resolution. In Eiffel [4], even this strategy is rejected because conflict-resolution will require knowing the details of how a given method is inherited, which is global information. Instead, whenever there is a fortuitous name-conflict due to multiple inheritance, the user is expected to use the renaming mechanism provided, to make available all the methods associated with the ancestors under different names.

Our approach to disambiguation takes a middle road. It is flexible in that it does not have an arbitrary built-in conflict-resolution strategy. Additionally, it enforces a certain discipline by imposing reasonable restrictions on the kinds of strategies that can be expressed. We illustrate this with an example shown in Figure 4. Let class  $c$  multiply inherit from classes  $e$  and  $f$  that are directly associated with a property  $p$ . Assume also that  $e$  and  $f$  conflict on  $p$ . The resulting inheritance conflict can be resolved by specifying which one of the two associations dominates. In case there is another class  $d$  that also multiply inherits from  $e$  and  $f$ , the inheritance conflict with respect to  $p$  for  $d$  is resolved

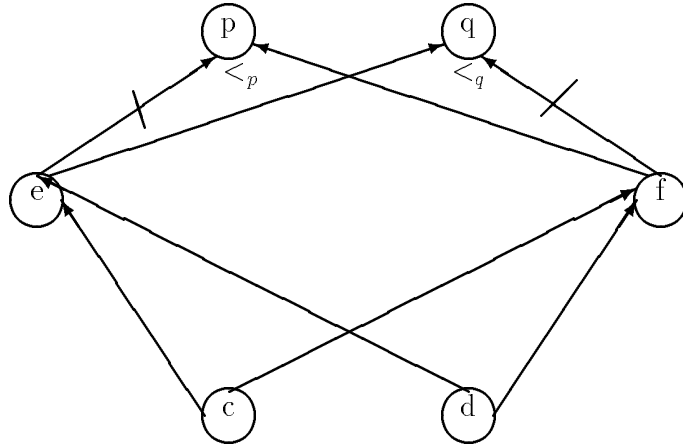


Figure 4: Preferential Inheritance

in exactly the same way as that for  $c$ . In addition, if classes  $e$  and  $f$  conflict on a different property  $q$ , we can specify a new dominance relation, different from the one used for  $p$ , to resolve the conflict.

In summary, the conflict resolution scheme we advocate is a function of a property and all the classes that are directly associated with it. Our scheme looks reasonable for specifying additional information to resolve ambiguity in inheritance networks. However, it may not be very satisfactory for object-oriented systems in general because the information about the child-parent link in the object hierarchy is with the child, and the definition of constants, instance variables and methods are all encapsulated in an “ancestor” class definition.

## 6 Conclusions

To summarize, both the upward and the downward views of inheritance are easy to specify for tree-structured hierarchies with exceptions. An efficient top-down algorithm can be designed to determine all properties that an individual inherits, and an efficient bottom-up algorithm can be given to compute all individuals that possess a given property based on the equivalence of the two approaches. However, in general, only the upward view can be formalized by designing local constraints on the meanings of the nodes in the network.

The downward view, in contrast, is inherently nonlocal and hence, difficult to formalize satisfactorily. We also argue that preferential inheritance provides a reasonable way of specifying additional conflict resolution information that can be integrated with the upward view of inheritance.

## References

- [1] H. Geffner, Default Reasoning: Causal and Conditional Theories, *Ph.D. Dissertation*, University of California at Los Angeles, 1989.
- [2] J. Horty, R. Thomason, and D. Touretzky, A skeptical theory of inheritance in nonmonotonic semantic networks, *Artificial Intelligence*, **42**, pp. 311-348, (1990).
- [3] J. Horty, A credulous theory of mixed inheritance, In *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, M. Lenzerini *et al* (eds.), John Wiley and Sons, 1991.
- [4] B. Meyer, Object-oriented Software Construction, Prentice Hall International, 1988.
- [5] Krishnaprasad Thirunarayan and M. Kifer, A theory of nonmonotonic inheritance based on annotated logic, *Artificial Intelligence*, **60**, pp. 23-50, (1993).
- [6] Krishnaprasad Thirunarayan, An analysis of property-flow view vs individual-flow view of inheritance, In *Proc. of the Sixth International Symposium on Methodologies for Intelligent Systems*, pp. 256-265, 1991.
- [7] T. Krishnaprasad, M. Kifer, and D. S. Warren, On the declarative semantics of inheritance networks, In *Proc. of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 1093-1098, 1989.
- [8] T. Krishnaprasad, The Semantics of Inheritance Networks, *Ph.D. Dissertation*, State University of New York at Stony Brook, 1989.

- [9] H. Przymusinska and M. Gelfond, Inheritance hierarchies and autoepistemic logic, Technical Report Computer Science Department, University of Texas at El Paso, 1989.
- [10] E. Sandewall, Nonmonotonic inference rules for multiple inheritance with exceptions, In *Proc. of the IEEE*, Vol. 74, No. 10, 1986, 1345-1353.
- [11] B. Selman and H.J. Levesque, The tractability of path-based inheritance, In *Proc. of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 1140-1145, 1989.
- [12] M. Stefik and D.G. Bobrow, Object-oriented programming: themes and variations, In *The AI Magazine*, pp. 40-62, 1986.
- [13] L. A. Stein, Resolving ambiguity in nonmonotonic inheritance hierarchies, *Artificial Intelligence*, **55**, pp. 259-310 (1992).
- [14] R. Thomason and J. Horty, Logics for inheritance theory, In *Non-Monotonic Reasoning*, M. Reinfrank *et al* (eds.), Springer-Verlag, 1989.
- [15] D. Touretzky, The Mathematics of Inheritance Systems, Morgan Kaufmann, Los Altos, 1986.
- [16] D. Touretzky, J. Horty, and R. Thomason, A clash of intuitions: the current state of nonmonotonic multiple inheritance systems, In *Proc. of the Tenth International Joint Conference on Artificial Intelligence*, pp. 476-482, 1987.