Nonmonotonic Inference Rules for Multiple Inheritance with Exceptions

ERIK SANDEWALL

The semantics of inheritance "hierarchies" with multiple inheritance and exceptions is discussed, and a partial semantics in terms of a number of structure types is defined. Previously proposed inference systems for inheritance with exceptions are discussed. A new and improved inference system is proposed, using a fixed number of nonmonotonic inference rules. The hierarchy is viewed as a set of atomic propositions using the two relations isa (subsumption) and nisa (nonsubsumption). General results concerning systems of nonmonotonic inference rules can immediately be applied to the proposed inference system.

I. Multiple Inheritance with Exceptions: Surprisingly Difficult

Inheritance "hierarchies" are a classical mechanism in artificial intelligence. They allow information to be stated about general concepts, and to be "inherited" by their specializations and instantiations, through one or more layers. Unlike the block structures, and other inheritance mechanisms in other branches of computer science outside AI, artificial intelligence research has often emphasized the need for multiple inheritance, where a more specific concept may inherit information from several more general concepts. For example, the concept "boy" should be able to inherit information both from "male person" and from "child," neither of which is a specialization of the other.

A second requirement on practical inheritance hierarchies is that they should allow for exceptions. For example, the concept "ostrich" is a specialization of "bird," and should inherit the properties of birds, except certain properties such as being able to fly.

Exceptions are fairly easy to deal with in single-inheritance systems, and can be obtained through conventional block structures. Multiple inheritance without exceptions is also easy to deal with theoretically. The combined structure, multiple inheritance with exceptions, offers, however, a number of unpleasant and challenging surprises. The purpose of the present section is to discuss them. In the sequel, we shall use the shorter term "inheritance systems" for our topic, since the precise term "systems with multiple inheritance and exceptions" is too elaborate.

Manuscript received May 6, 1985; revised May 6, 1986. This research was supported by the Swedish Board of Technical Development.

The author is with the Department of Computer and Information Science, Linköping University, 58183 Linköping, Sweden.

We shall represent exceptions through separate exception links in the structure, as previously used by Fahlman [2], Etherington and Reiter [1], and Touretzky [7]. One then has a structure of nodes, and two kinds of links between nodes, which will here be written *isa*(*x*, *y*) and *nisa*(*x*, *y*). The intended meaning is "any *x* is usually a *y*" and "any *x* is usually not a *y*," respectively. The inclusion of "usually" means that those statements may be qualified by other links in the hierarchy. For example, some of the peculiar properties of ostriches and penguins, relative to other birds, may be expressed as follows:

isa(Ostrich, Bird)
isa(Penguin, Bird)
isa(Bird, AbleToFly)
nisa(Ostrich, AbleToFly)
nisa(Bird, UprightWalker)
isa(Penguin, UprightWalker)

Fig. 1 illustrates this structure, and it also illustrates the graphical notation that we shall use in the sequel (and which

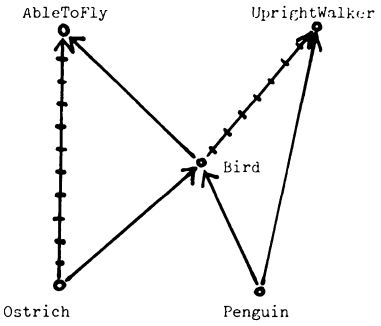

Fig. 1.

is standard in the literature); namely, to represent *isa* by an ordinary arrow and *nisa* by an arrow with crossbars.

The primary use of such a structure is to "answer questions," or more specifically, for answering questions about explicit and implicit (derivable) links between nodes. If *I* is an inheritance structure, then an extension of *I* is a larger structure which contains the same nodes and links as *I*, but where some additional links may have been added; namely, all those links that can be derived from the links in *I*. (No

0018-9219/86/1000-1345$01.00 © 1986 IEEE

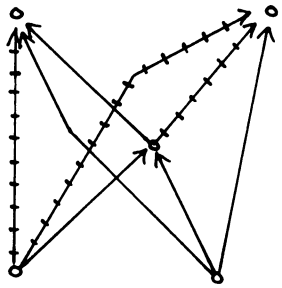

Fig. 2.

extra nodes may be added.) Fig. 2 shows the extension of the structure in Fig. 1.

We shall now consider three inheritance structures which are particularly interesting, and which we shall refer to as Type 1, Type 2, and Type 3 structures. Each of them is a configuration of a few nodes and arcs, so a practical inheritance structure may contain one or more substructures of each type.

The following often-used example, also shown in Fig. 3(a), is a Type 1 structure:

isa(Clyde, RoyalElephant)
isa(RoyalElephant, Elephant)
isa(Elephant, Gray)
nisa(RoyalElephant, Gray)

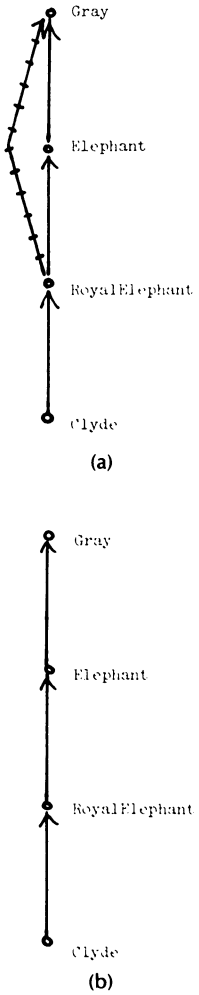

Fig. 3. (a) Type 1.

In this example, we wish to be able to infer

nisa(Clyde, Gray)

because he is a *RoyalElephant*, which are known to be non-*Gray*. Thus the *nisa* link from *RoyalElephant* to *Gray* inhibits the transitivity of *isa* links, which would otherwise imply

isa(RoyalElephant, Gray)

causing a contradiction (or implying that there are no *RoyalElephants*). Furthermore, we wish to be able to infer, in this example,

isa(Clyde, Elephant)

but still the transitive conclusion from that statement plus

isa(Elephant, Gray)

should be blocked.

The particular problem with Type 1 structures is that they introduce nonmonotonicity. Consider the structure in Fig. 3(b), which is a part of Fig. 3(a). The extension of Fig. 3(b) naturally contains the link

isa(Clyde, Gray)

Thus as we go from the structure in Fig. 3(b), to the larger structure in Fig. 3(a), some of the possible conclusions are lost. We say that a logical system is monotonic if it satisfies

$$A \subseteq B \rightarrow Th(A) \subseteq Th(B)$$

where A and B are sets of propositions, and $Th(A)$ is the set of conclusions that can be obtained from A, i.e., the extension. If in our case we let A and B be sets of links, then Type 1 structures fail to satisfy the property of monotonicity.

The following is the standard example of a Type 2 structure, also illustrated in Fig. 4:

isa(Nixon, Republican)
isa(Nixon, Quaker)
isa(Quaker, Pacifist)
nisa(Republican, Pacifist)

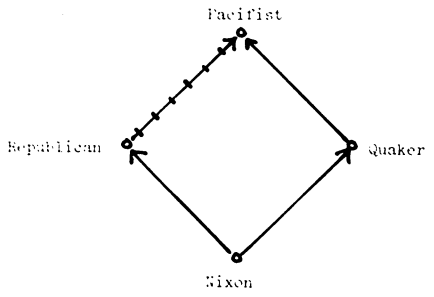

Fig. 4. Type 2.

The puzzling question is of course: is Nixon a Pacifist or is he not? The stance that one usually takes is that this structure has two distinct extensions, as shown in Fig. 5(a) and (b). In one extension, Nixon is a Pacifist, and in the other extension Nixon is not a Pacifist.

If such a structure arises in a practical situation, one would, of course, expect the software to look for additional information which allows it to choose between those two alternatives. But as long as we only study the inheritance

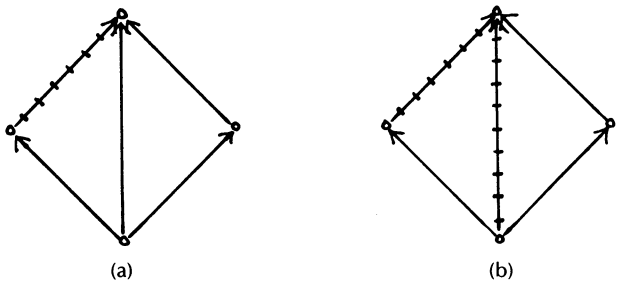

Fig. 5.

structures themselves, the additional information is not available, and the only possible approaches are to allow both alternatives, or to ban such structures altogether.

Although there are two possible extensions, it is not permitted to mix them arbitrarily. For example, if the original structure also contains the link

isa(RichardNixon, Nixon)

which makes sense if the node "*Nixon*" refers to any member of the Nixon family, then the first extension would contain the link

isa(RichardNixon, Pacifist)

and the second extension would instead contain the link

nisa(RichardNixon, Pacifist).

Fig. 6(a) and (b) shows the revised extensions. Notice how the relation between the nodes *RichardNixon* and *Pacifist*

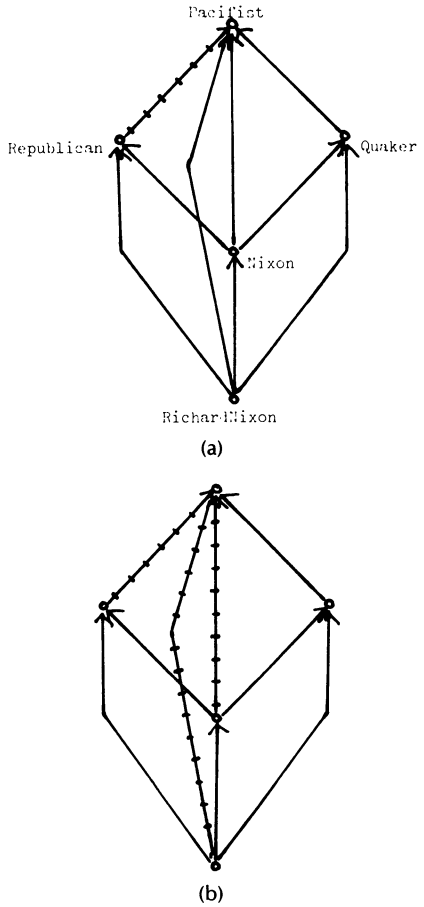

Fig. 6.

are obtained from a Type 1 structure in both cases. An incorrectly designed inference system would produce four extensions, including one where *Nixon* is a *Pacifist* and *RichardNixon* is not, and also the reversed one. Touretzky uses the term decoupling for that anomaly.

It should not come as a surprise that multiple extensions arise in the same context as nonmonotonicity. Sandewall [4] describes the nonmonotonic operator Unless, and the observation that it leads to multiple extensions.

The following is a Type 3 structure, also illustrated in Fig. 7:

isa(Whale, Mammal)
isa(Mammal, LandAnimal)
nisa(Whale, LandAnimal)
isa(LandAnimal, x).

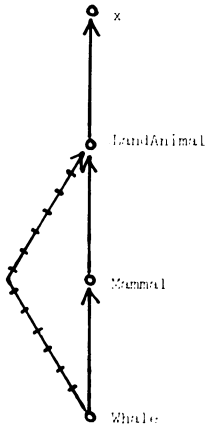

Fig. 7. Type 3.

The issue is whether the link

isa(Whale, x)

should be allowed in the extension(s) of this structure. It seems natural that the answer must be "no," since the only reason why a *Whale* would be an *x* is by the three-step *isa* chain, and the *nisa* link specifies an exception to its lower two links. However, if the link

isa(Mammal, x)

is added to the original structure as an independent fact, then there does not seem to be any reason for not drawing the conclusion that *Whales* are *x*'s. This means that different facts have a different derivational "power." The distinction is not simply between axioms and derived facts, however. Suppose that the two links

isa(Mammal, y)
isa(y, x)

are both added, then again one should be allowed to conclude that *Whales* are *x*'s. The difference in derivational "power" is, therefore, made on the basis of derivational dependency, or which links are derived from, and therefore dependent on which other links.

The following example also illustrates this point. Suppose we augment the "*Nixon*" example of Type 2 structures, with the additional link

isa(Pacifist, DraftEvader)

as in Fig. 8. Like before, we have two choices for the relation

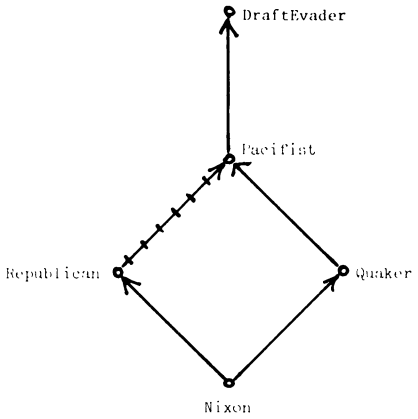

Fig. 8. Type 3B.

between the *Nixon* and *Pacifist* concepts, as shown in Figs. 9(a) and 10(a). The corresponding extensions are shown in Figs. 9(b) and 10(b). The case of Fig. 9(b) offers no surprises, but in Fig. 10(b) both the following links are derived ones:

nisa(Nixon, Pacifist)
isa(Quaker, DraftEvader)

and if those links would have equal status, then one should be allowed to derive

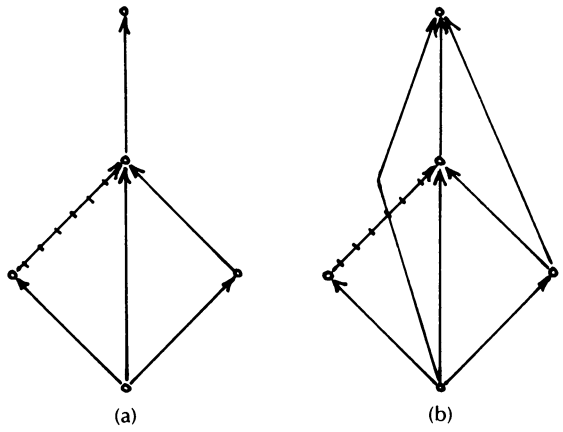

Fig. 9.

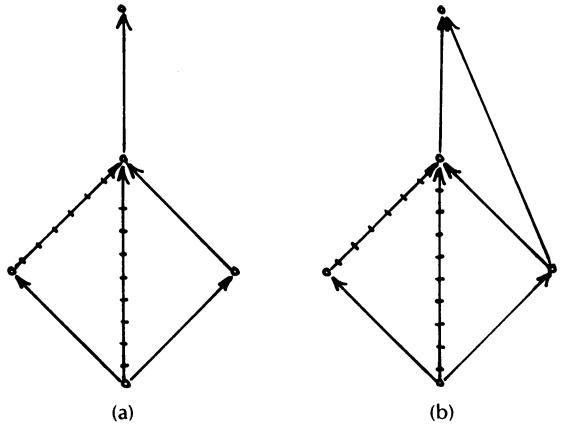

Fig. 10. (b) Type 3B.

isa(Nixon, DraftEvader)

but it would seem unsatisfactory to accept that conclusion, since it relies on the existence of an implicit link

isa(Nixon, Pacifist)

which has been rejected by the choice of a *nisa* link instead between those two nodes. We refer to this structure as Type 3B.

In the context of derivational history, we must also consider the structure shown in Fig. 11, which we shall refer

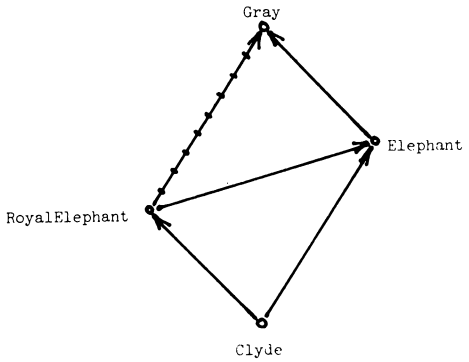

Fig. 11. Type 1B.

to as Type 1B. It is like Type 1, except that the link

isa(Clyde, Elephant)

in the example is now an axiom rather than a derived link. We require that also in this case there shall be only one extension, where *Clyde* is non-*Gray*, just like in Type 1.

Type 1B is also similar to Type 2 structures, except that in Type 1B we have the extra "diagonal" link, namely

isa(RoyalElephant, Elephant)

in the example. It is this extra link that restricts one of the two extensions that Type 2 structures have.

Finally, we give in Fig. 12 a slightly more complex example, which does not introduce any new situation type,

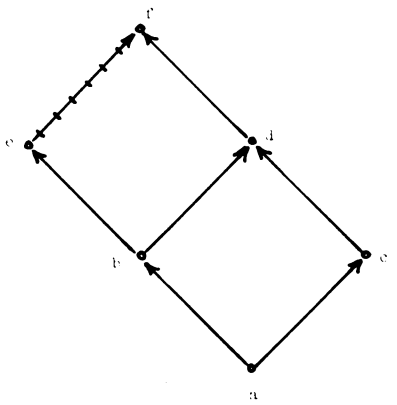

Fig. 12.

but which illustrates the applications of the given three types. (The representation by formulas is omitted.) Fig. 13(a) and (b) shows structures which must certainly be admitted

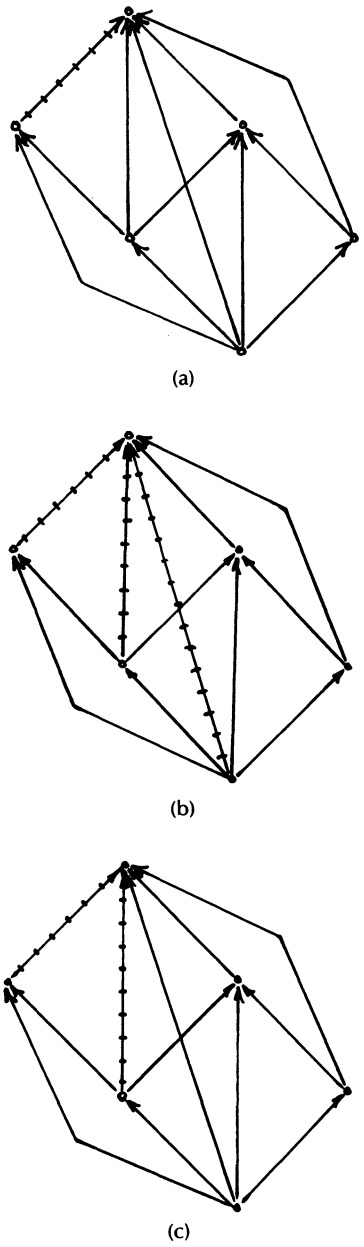

Fig. 13.

as extensions. But what about the structure in Fig. 13(c)? The link *isa(a, f)* may not be derived from *isa(a, d)* and *isa(d, f)*, since we have a Type 1 substructure with *a, b, d, f*. It could, however, be obtained from *isa(a, c)* and *isa(c, f)*, since the nodes *a, b, c, f* form a Type 2 substructure.

At this point it is clear that the structures of multiple inheritance with exceptions have a very nonstandard logic. Their analysis requires the use of concepts such as "conclusions" and "derivation," but the three types of special structures all run counter to the traditional rules and intuitions of logic. Type 1 structures show that we have nonmonotonicity; Type 2 structures show that we have to deal with multiple extensions; and Type 3 structures show how the logical dependency between propositions affect the possible conclusions.

Before we proceed to the formal treatment of these structures, we must, therefore, discuss in some more depth how they relate to ordinary logic.

II. Philosophical Aspects of Inheritance Systems

The concept of having several extensions in a logical system is an unusual one, and may be considered hard to accept. That depends, however, on what we think of as the goal for logic. If the goal is to formulate what are the admissible conclusions, one at a time, from a given set of axioms, then one will expect to have a unique extension; namely, the set of all (individually) admissible conclusions. If, on the other hand, one takes as the goal to specify what are admissible belief structures, when a certain set of "base beliefs" is given, then one should not be so disturbed by the existence of several distinct, but admissible belief structures or extensions. Thus the quest for monotonicity is related to the yearning for the single truth.

We proceed now to the issue of semantics. In order to develop a logical theory, one must first identify the structure of the phenomena that one wishes to describe, and then define the language and inference machinery of the logic, i.e., the syntax of formulas, the choice of axioms and inference rules, and so forth. The semantics of the logical theory is then the formal expression of the "structure of the phenomena," together with an account of how it relates to the selected language.

In the case of inheritance systems, it is not so easy to define semantics. One might like to base semantics on the notion of individual objects, which are "members of" or "included in" the concepts represented by nodes in the inheritance structure. Then *isa(x, y)* would mean something along the lines of "all *x* are *y*" or, since exceptions are allowed, "most *x* are *y*." Exactly what does "most" mean? Suppose 90 percent is sufficient. But then if we have

$$isa(x_0, x_1),\ isa(x_1, x_2),\ \ldots,\ isa(x_9, x_{10})$$

we might in the worst case obtain that only about 35 percent of the x_0 are x_{10}!

Another annoying problem is that when we analyze fine points of the proposed system (e.g., whether to admit the extension in Fig. 13(c)), proposed solutions cannot be refuted by concrete counter-examples, exactly because it is the purpose of the system to accomodate occasional counter-examples. Consider the example of a Type 3 structure that was given above, where the issue was whether

$$isa(Whale, x)$$

should be an admitted conclusion. If we select

$$x = AnimalWithLegs$$

then all the links in the structure are in accordance with our common sense knowledge, and *isa(Whale, x)* should not follow. If, on the other hand, we select

$$x = AnimalThatObtainsOxygenFromAtmosphere$$

then again all the links in the structure agree with common sense knowledge, and it would be correct for *isa(Whale, x)* to follow. (We cannot choose $x = AnimalWithLungs$ since insects do not have lungs.) Regardless whether *isa(Whale, x)* is in the extension or not, we can accomodate both choices of *x* in an application, simply by adding an extra *isa* or an extra *nisa* link, for the case where the deduction machinery does not yield the desired result.

The task of defining an inheritance system might, therefore, be thought of as a design task, rather than as the task

of identifying the correct forms of reasoning. The system should be designed so that reasonable consistency is obtained, and so that one minimizes the number of extra *isa* or *nisa* links that have to be introduced in order to adapt the results of the inference machinery to reality.

The published attempts to define the formal properties of inheritance systems have, therefore, used a nonstandard inference process for defining extensions. One then defines an inference operator that will add one more link (or other, similar construct) to an inheritance structure. The admissible extensions of an inheritance structure I are defined to be those supersets of I which are closed with respect to the inference operator (i.e., everything that the operator would add to the closure, is already a member there), and which are well founded (i.e., all members of the extension must either be members of I, or have been obtained through the inference process).

By the standards of traditional logic, it is debatable whether such an inference process qualifies as a satisfactory semantics. One would like to have a semantics which defines the underlying meaning, and which the inference process could be measured against. But as we have already seen, such an underlying meaning is hard to identify and make precise. Inheritance systems were introduced because of their utility, not because of elegant formal properties.

Another, although related, problem is that the inference processes for inheritance systems that have been proposed so far, are not particularly transparent. The reader is not likely to accept them as they stand, and in fact the only reasonable way to approach them is through specific cases, like our Type 1, 2, and 3 structures. I, therefore, propose that we consider such collections of structure types as the definition of the semantics, for the time being. They do satisfy two basic requirements for a semantics: they can easily be grasped intuitively, and one can effectively determine whether a proposed inference process will deal with them correctly. The remaining problem is that they are only partial semantics, so it would be possible to have two inference processes, both of which agree with the semantics, i.e., they treat the specified structure types correctly, but still they can be proven to be nonequivalent. If and when that happens, one will have to identify additional structure types for which those processes differ, and add them to the semantics, thereby making it more precise.

If we think of logic and/or information science as an empirical endeavor, where information structures which actually occur in the real world are identified and formally characterized, then such an iterative strategy should be quite acceptable.

III. Current Theories for Inheritance Systems

A. Etherington and Reiter: Default Logic

Etherington and Reiter [1] express each link in the inheritance structure as an inference rule in default logic [3]. They give criteria and proofs for the existence of extensions, and prove the correctness of certain inference algorithms. They claim that the resulting system defines a semantics for inheritance systems.

Their approach, however, suffers from the problem that the transformation from links to inference rules is not a local operation. In the "Clyde" example for Type 1 structures, for instance, the link that we represented as

isa(Elephant, Gray)

would have to be rewritten as

if Elephant(x)
and unless it leads to a contradiction to assume
Gray(x) ∧ ¬ RoyalElephant(x)
then Gray(x)

As Touretzky [6] pointed out, the need to write exceptional cases explicitly into the inference rules that may be affected by them, is very impractical. Also, of course, the very point with nonmonotonic reasoning and exception links is that we should not have to perform that chore.

In order to deal with Type 3 structures, where derivational dependency comes into play, one would presumably have to introduce additional artifacts into the inference machinery.

B. Touretzky's Theory for Inheritance Systems

Touretzky [6], [7], uses an approach which is more remote from ordinary logic: he considers an inheritance system as a set of paths, where each path is an annotated sequence of nodes. A link is a path of length 2, and in the originally given structure all paths are links. The inference operator combines two paths of length k into one new path of length $k + 1$. Therefore, unlike ordinary logic, each derived path contains its entire derivation history. This allows a correct treatment of Type 3 structures. Also, although the inference operator primarily operates on two existing paths, it may be inhibited by other paths in the structure. This allows a correct treatment of Type 1 and Type 2 structures as well.

The following summarizes the concepts and notation that are used in [7, ch. 2], together with some additional notation that we need.

We consider sequences $\langle u_1, \ldots, u_n \rangle$, called paths, where each u_i has either of the following forms

$+x_i$
$-x_i$

where again each x_i is a node in the network, and where the sign must be + in each u_i, except u_n. (Minus signs in other positions would be used, e.g., for "non-x objects are y's," expressed as

$$\langle -x, +y \rangle$$

but Touretzky [7] does not analyze such constructs.)

Touretzky [7] also allows u_i of the form

$$\#x_i$$

but we ignore that case here since it does not add to the difficulty of the analysis. A path characterizes the conjunction of the following propositions:

$isa(x_1, x_2)$
$isa(x_2, x_3)$
$\ldots$
$isa(x_{n-1}, x_n)$, *if* $u_n = +x_n$
$nisa(x_{n-1}, x_n)$, *if* $u_n = -x_n$.

We write $u \pm v$ when u and v contain the same node but with opposite signs. (Thus $\pm$ is a relation symbol here.) If

Φ is a set of paths, $C(\Phi)$ is defined by

$$C(\Phi) = \{\langle u_1, u_n\rangle \mid \langle u_1, u_2, \ldots, u_n\rangle \in \Phi\}.$$

A path $\sigma = \langle u_1, u_2, \ldots, u_{n-1}, u_n\rangle$, where $n > 2$, is said to be inheritable in a set Φ of paths iff the following holds:

$\langle u_1, \ldots, u_{n-1}\rangle \in \Phi$
$\langle u_2, \ldots, u_n\rangle \in \Phi$
Φ does not contradict σ
Φ does not preclude σ

where the last two conditions are defined as follows:

Φ contradicts σ iff some $\langle u_1, v\rangle \in C(\Phi)$
where $v \pm u_i$ for some i.
Φ precludes σ iff some $\langle v, w\rangle \in \Phi$ where $w \pm u_k$
for some k, where
$1 < k \leq n$, and either
$v = u_i$ for some $i \leq k$, or:
$\langle u_1, \ldots, u_i, v_1, \ldots, v_m, u_{i+1}\rangle \in \Phi$
and $v = v_j$ for some j
and $1 \leq i < k$.

Actually, Touretzky [7] first introduces this definition with the stronger restriction $k = n$, and then relaxes it to the above formulation. He hypothesises that the two variants result in the same extensions.

The set Φ of paths is a grounded expansion of a set S iff:

$S \subseteq \Phi$
Φ is closed under inheritance
every path in $\Phi - S$ is inheritable in Φ.

This is what we have called an extension above, and we shall use the two terms synonymously.

Since an inherited path is longer than the two paths that were used to form it, for every grounded expansion Φ of S there must be some sequence of sets

$$S_0, S_1, \ldots, S_m$$

where $S = S_0$, $S_m = \Phi$, and each S_i is obtained from its predecessor by adding one or more paths that is inheritable in S_{i-1}.

It is easy to verify that this definition treats our structure types correctly. In Type 1 structures there are two potential and contradictory derived links (*Clyde* is *Gray* or non-*Gray*, in the example), and one of them has to be inhibited. This is done through the preclusion condition. The two "either" cases in the definition of preclusion deal with Type 1 and Type 1B, respectively.

In Type 2 structures, the two extensions are again obtained through the preclusion condition, which guarantees that whichever of the competing links is derived first, that will inhibit the other.

Type 3 structures, finally, are handled correctly by virtue of the contradiction requirement. For Type 3B, the last clause in the definition of a grounded expansion plays a key role for avoiding the incorrect conclusion

isa(Nixon, DraftEvader)

in the example in Fig. 10(b). That proposition is, in fact, derivable in some subsets of the grounded expansion, but it is not rederivable in the full expansion.

As we have already discussed, it is difficult to understand the full consequences of Touretzky's definitions. For example, although Type 1B is handled correctly, the slightly

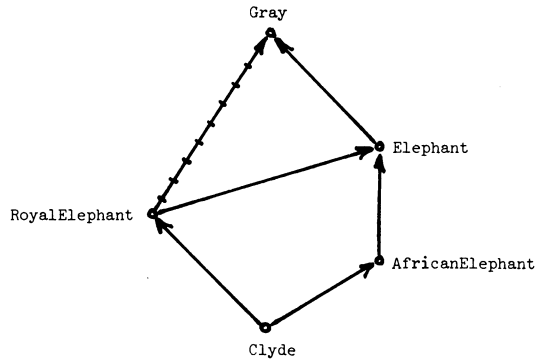

Fig. 14. Type 1C.

modified structure shown in Fig. 14 will result in two extensions, i.e., one where *Clyde* is *Gray* and one where he is non-*Gray*. The reason is that the definition of preclusion only allows the sequence of nodes $v_1, \ldots, v_m$ to be inserted between two successive nodes u_i and u_{i+1}, so the extra node "*AfricanElephant*" in Fig. 14 results in a loss of the preclusion condition. We refer to this structure as Type 1C.

This apparent anomaly supports our suggestion that the collection of structure types should be seen as the currently available, partial semantics for inheritance systems. New, proposed theories for inheritance should first be calibrated against that semantics, and only then should one maybe investigate whether they are equivalent to Touretzky's or other existing theories.

IV. An Alternative Inference Machinery, Using Nonmonotonic Logic, for Multiple Inheritance with Exception

Touretzky chose to use paths, or sequences of nodes, as the "propositions" in his inference machinery. This allows him to capture the special properties of the structure types, but it disassociates him from ordinary logic, which makes it particularly difficult to extend the system, or combine it with other logic. Etherington and Reiter, on the other hand, stayed within a nonmonotonic logic with considerable generality, but at the expense of sacrificing the spirit of exception statements.

In this section we shall show that it is possible to obtain the best of both worlds. We use propositions of the following forms:

isax(x, y, s)
isa(x, y, s)
precl(x, y, z, s)
cntr(x, y, z, s)

where x, y, and z are nodes, and s is either of the symbols $+$ or $-$. The ternary *isa* relation is defined by

$$isa(x, y, +) = isa(x, y)$$
$$isa(x, y, -) = nisa(x, y).$$

If we wish to stay within a single-sorted logic we can, of course, view ternary *isa* as an abbreviation. Type 3 structures, and the dependency conditions they imply, are handled by the separate relation *isax* which is a stronger form of *isa*. Thus *isax(x, y, s)* means that *isa(x, y, s)* independently of other links. In practice one would use *isax* for stating the explicitly given facts, or axioms (hence the relation's name),

or at least for stating those explicitly given facts that are known to be independent of all others. The relation *isa* is then used for both given and derived links. This machinery is counterintuitive, since it makes an irrelevant distinction—between given and derived facts—and fails to make the relevant distinction of when there are dependencies between facts. However, it does work, since independent, derived facts are treated in a different way, and as we do not view the inference machinery as a way of clarifying the semantics we do not have any qualms about it.

The relations *precl* and *cntr* are used for technical purposes, which will be apparent in the inference rules. They are vaguely related to Touretzky's preclusion and contradiction conditions, respectively.

If s is $+$ or $-$, then $-s$ is the opposite sign as s.

Inference rules are written in the general form "If Φ_1 is known, and Φ_2 is not known, then infer Φ_3." As we showed in [5] one may determine extensions for such rules as follows: Construct a sequence of successively increasing sets of propositions,

$$\Gamma_0, \Gamma_1, \Gamma_2, \ldots, \Gamma_n, \ldots$$

where each Γ_{i+1} is constructed from Γ_i by selecting an instantiation of a rule where Φ_1 is a subset of Γ_i, and Φ_2 is disjoint from Γ_i, and then choosing Γ_{i+1} as $\Gamma_i \cup \Phi_3$. The process is continued to its (possibly infinite) limit, where we obtain an extension. Depending on the order in which the rules are applied, we may obtain several different extensions. Among them, all consistent extensions are accepted, but not those extensions containing both the propositions p and $\neg p$ for some p.

Intuitively speaking, this makes it possible for an inference rule to generate "mines" which will later on create a contradiction in the proof sequence, and thereby render it invalid. The relations *precl* and *cntr* are used for exactly this purpose.

The following is our set of inference rules:

1) if *isax(x, y, s)* $\in \Gamma$
   then add to Γ:
   isa(x, y, s)
2) if *isa(x, y, s)* $\in \Gamma$
   then add to Γ:
   $\neg$ *isa(x, y, −s)*
3) if the following are members of Γ:
   isa(x, y, +)
   isa(y, z, s)
   and the following are not members of Γ:
   isa(x, z, −s)
   cntr(x, y, z, s)
   then add to Γ:
   isa(x, z, s)
   precl(x, y, z, s)
   $\neg$ *cntr(x, y, z, s)*
4) if the following are members of Γ:
   precl(x, y, z, s)
   isa(x, v, +)
   isa(v, y, +)
   then add to Γ:
   precl(x, v, z, s)
5) if the following are members of Γ:
   precl(x, y, z, s)
   isa(y, z, −s)
   then add to Γ:
   $\neg$ *isa(y, z, −s)*
6) if the following are members of Γ:
   isa(x, y, +)
   isa(y, z, +)
   isa(x, z, −)
   isa(z, v, s)
   and the following is not a member of Γ:
   isax(y, v, s)
   then add to Γ:
   cntr(x, y, v, s).

In order to understand how these rules work, let us first ignore Rule 6 and the *cntr* literals in Rule 3. The thus simplified rules are sufficient for dealing with Type 1 and Type 2 structures, and their variants.

Rule 3 is set up so that it will not produce combinations of the form

isa(x, y, +) and *isa(x, y, −)*

which would immediately lead to a contradiction through Rule 2. Through Rule 3 we, therefore, obtain a correct treatment of Type 2 structures. Also, when Rule 3 is applied, the auxiliary proposition

precl(x, y, z, s)

is obtained. Rule 4 then "slides" the second argument of the *precl* proposition down through the structure, which is used for spotting Type 1, 1B, and 1C structures. Rule 5 is used to invalidate undesirable extensions. (This is a counterpart of backtracking, in the logical system.)

Consider, for example, the contents of Fig. 3(a), which would be represented in our notation as follows:

isa(Clyde, RoyalElephant, +)
isa(RoyalElephant, Elephant, +)
isa(Elephant, Gray, +)
isa(RoyalElephant, Gray, −).

There are two main alternatives for the order in which the inference rules are applied. If we start by inferring that *Clyde* is non-*Gray*, we obtain using Rule 3:

isa(Clyde, Gray, −)
precl(Clyde, RoyalElephant, Gray, −).

After that we also conclude that Clyde is an elephant, and obtain:

isa(Clyde, Elephant, +)
precl(Clyde, RoyalElephant, Elephant, +)

An attempt to deduce that *Clyde* is *Gray* at this point is prevented by the inhibiting conditions in Rule 3, since we have already concluded that he is non-*Gray*.

In the other main alternative, we do not first conclude that *Clyde* is non-*Gray*. Instead we start with

isa(Clyde, Elephant, +)
precl(Clyde, RoyalElephant, Elephant, +)

using Rule 3, and then

isa(Clyde, Gray, +)
precl(Clyde, Elephant, Gray, +).

At this point we cannot conclude that *Clyde* is non-*Gray*, because of the symmetrical character of Rule 3. However,

Rule 4 allows us to conclude

precl(Clyde, RoyalElephant, Gray, +)

and then Rule 5 identifies the problem and reacts by intentionally creating a contradiction, namely

$\neg$ *isa(RoyalElephant, Gray, −)*

which contradicts what was originally stated as

isa(RoyalElephant, Gray, −).

In summary, two extensions are created, but one of them develops an inconsistency, and only the other one is admissible. For simplicity, in this example we have started from the *isa* propositions, and ignored the original step from *isax* to *isa*.

This machinery for Type 1 structures also prevents decoupling.

For Type 3 structures, we need the full set of rules. Rule 6 identifies the Type 3 structure. For example, in Fig. 7, we would have x = *Whale*, y = *Mammal*, z = *LandAnimal*, and $v = x$ in Rule 6. Also Rule 6 can be applied if *isa(Mammal, x)* is only a derived link, but not if it is a given link, because in the latter case there will be an *isax* proposition. The conclusion from Rule 6,

cntr(x, y, v, s)

indicates that it is not allowed to combine

isa(x, y, +)
isa(y, v, s)

using Rule 3. Rule 3, therefore, checks for this condition in a nonmonotonic assumption, so if Rule 6 has been used then Rule 3 will not be applied. On the other hand, if Rule 3 has been applied first, then Rule 6 will be applied anyway, and a contradiction will occur which will invalidate the present derivation.

If the "bypass link" *isa(Mammal, x)* is derived but independent, because there is some separate node *r* for which *isa(Mammal, r)* and *isa(r, x)*, then it will not be possible to chain *isa(Whale, Mammal)* with *isa(Mammal, x)*, to obtain the conclusion *isa(Whale, x)* that we should be allowed to have in this case. However, the desired conclusion can be obtained in another order, namely

isa(Whale, Mammal)
isa(Whale, r)
isa(Whale, x)

without being inhibited by Rule 6.

In summary, the proposed inference machinery handles the three structure types and their variants correctly. It is clearly not equivalent to Touretzky's machinery since they handle Type 1C differently. It would probably not lead to any complications to change his system in that respect, but we have not tried to prove the equivalence between a revision of his system and our system.

It is certainly too early to claim that the present system, or any other proposed inference machinery, deals correctly with all situation types that one can think of. (In particular, the literature rarely deals with the case of circular *isa* structures, and they are also not represented in the structure types in this paper.) In fact, it is not even clear that there will always be a good common-sense answer as to what is the correct way of dealing with a situation type. The present contribution is only claimed to be a step in the iterative discovery process that we must pursue at present. Meanwhile, it has also pointed towards a way of accomodating the peculiar characteristics of inheritance systems semantics in nonmonotonic logic, for which some results at least are known. Thus it is not necessary to use the inheritance paths, containing the trace of the derivation, as Touretzky does.

In a previous paper about nonmonotonic logic [5], we proved results for systems of nonmonotonic inference rules which are directly applicable here. The following results follow directly, with the requirement on extensions that they must be consistent as we have assumed in this section:

Proposition: Let Φ and Φ' be two extensions of an inheritance structure S, for which $\Phi \subseteq \Phi'$. Then $\Phi = \Phi'$.

Proposition. Every union of distinct extensions of an inheritance structure, is inconsistent.

Touretzky proved these results for his system, but for the latter proposition his proof only applies if the *isa* relation in S is acyclic. By relying on nonmonotonic logic we obtain the same results as special cases of generally applicable theorems, and we obtain a more general result since we do not need the limitation of an acyclic *isa* relation.

References

[1] D. W. Etherington and R. Reiter, "On inheritance hierarchies with exceptions," in *Proc. (U.S.) Nat. Conf. on Artificial Intelligence*, pp. 104–108, 1983.

[2] S. E. Fahlman, *NETL: A System for Representing and Using Real-World Knowledge*. Cambridge, MA: MIT Press, 1979.

[3] R. Reiter, "A logic for default reasoning," *Artificial Intell.*, vol. 13, no. 1, pp. 81–132, 1980.

[4] E. Sandewall, "An approach to the frame problem, and its implementation," in B. Meltzer and D. Michie, Eds., *Machine Intelligence 7*. New York, NY: Wiley, 1972, pp. 195–204.

[5] ——, "A functional approach to non-monotonic logic," in *Proc. 1985 Int. Joint. Conf. on Artificial Intelligence*, pp. 00–00, 1985, and *Computat. Intell.*, vol. 1, no. 2, pp. 00–00, 1985.

[6] D. S. Touretzky, "Implicit ordering of defaults in inheritance systems," in *Proc. (U.S.) Nat. Conf. on Artificial Intelligence*, pp. 322–325, 1984.

[7] ——, "The mathematics of inheritance systems," Rep. CMU-CS-84-136, Dep. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, 1984; also London, UK: Pitman, 1986.