Negative Reasoning Using Inheritance

Lin Padgham
Department of Computer and Information Science
Linkoping University
S-581 83 Linkoping, Sweden
lin@ida.liu.se

Abstract

This paper presents methods of default reasoning which allow us to draw negative conclusions that are not available in some of the models for inheritance reasoning. Some of these negative conclusions are shown to be logically required, while others result from an extension of the model to include the notion of a default negative assumption. The negative default assumption operator is exactly symmetrical to positive default assumption, and supports the drawing of extra negative conclusions. It is argued that in some domains negative conclusions are extremely important. An example is given from the medical domain to illustrate the usefulness of techniques for deducing negative facts. A formal definition of the inheritance model used, which in an earlier paper by the same author [Padgham 88] was shown to resolve a number of the classical problems in the literature on inheritance reasoning, is also given.

1. Introduction

Most of the literature on inheritance reasoning [Etherington 87, Fahlman 79, Touretzky 86, Sandewall 86, etc.] focusses on methods for collecting inherited information regarding an entity, in the presence of conflicting information. Conflicting information is usually of the type X is a Y, and X is not a Y.

This negative information regarding what X is not is often extremely important. For example in medical diagnosis, it is often equally important to rule out certain diseases as it is to obtain a positive diagnosis. Similarly if one is reasoning about prescriptions for medical drugs, one is extremely interested in the negative information, i.e. contra-indications, or incompatibilities. It is often just this negative information which is not quite so immediate and may need to be deduced, using in part an inheritance reasoning mechanism.

We present a method for doing default reasoning with an inheritance schema which allows both default and strict reasoning about negative conclusions in the same way as it does about positive conclusions. In order to present the negative reasoning we need first to present the basic model, and the reasoning methods for positive and explicit negative information. Deduction of extra negative information then follows from these. We will illustrate with some examples which are a little more complex than the usual examples found in the literature because one needs the added complexity in order to benefit from the added reasoning. We will also attempt to relate the methods to some standard examples, in order to at least explain the results within a familiar context. However the real power of these mechanisms will be seen in more complex real-world reasoning systems, for which the methods are of course intended.

2. Overview of The Inheritance Model

The basic model is that both objects and types can be described by sets of characteristics, which can be partially ordered according to the notion of more information. Thus for an object to be a member of a particular class/type it must contain as much information as is required for the type, or more; that is it must contain at least all the characteristics required by the type. Similarly a type A is a subtype of type B if A contains all the characteristics of B, plus some extra characteristics.

Formally types are points in a lattice of *descriptors* within which a partial order $\sqsupseteq$ and lattice operations $\sqcup$ and $\sqcap$ are defined. In the simplest case we assume C to be the set of all characteristics, choose

This work was sponsored by the Swedish Board for Technical Development.

descriptors as subsets of C, define A $\sqsupseteq$ B as $B \subset A$, and define $\sqcup$ and $\sqcap$ as $\cup$ and $\cap$ operations, respectively.

We introduce the notion that a type is defined by two descriptors, the *type core* and the *type default* The type default is the set of characteristics we would expect to find in a typical object of that type. The type core is a subset of the type default and consists of those characteristics strictly necessary for an object to be of that type. We introduce the notation X_d and X_c to refer to the default and core of X respectively. It is unimportant whether we can in fact enumerate all the characteristics in these two sets, the important point is that theoretically there are two such sets for every type. We can then have and reason about the partial information which is available concerning the descriptors and the relationships between them.

As indicated above, descriptors can be compared with one another on the basis of the more information relation $\sqsupseteq$, within the lattice formed by all possible subsets of C. (It should be stressed that this lattice is a theoretical entity and need never be enumerated). If a descriptor A $\sqsupseteq$ a descriptor B, this is equivalent to the statement that A *is a* B.

We note that $A_d \sqsupseteq A_c$. There may be many possibilities between the core and the default of the type which fulfill the core requirements but are in various ways deviant from the default. There may be some characteristics which seem to belong in the core descriptor of a type, rather than in the default descriptor, which nevertheless can be absent from particular individuals. For instance we would want to say that four-leggedness is a core characteristic for dogs. However there are certainly dogs who have lost a leg. Our position here is that individuals must always be allowed to be declared as members of a class even if they do not fulfill the core characteristics for that class. However no subclass may exist which does not fulfill its parent class' core characteristics. The fact that three-legged dogs exist does not warrant the creation of a class of such. If in some application (e.g. database for amputation section of a veterinary hospital) one wanted such a class, then the four leggedness property should be removed from the type core.

Because we have both core and default descriptors for each type we can talk about the $\sqsupseteq$ relation to and from both the core and the default points for each type. This allows us to make the statement that A is a B with the 4 following nuances:

* A's are always typical B's $(A_c \sqsupseteq B_d)$
* **A's are always B's (but not necessarily typical B's) $(A_c \sqsupseteq B_c)$**
* **A's are usually typical B's $(A_d \sqsupseteq B_d)$**
* **A's are usually B's (but not necessarily typical B's) $(A_d \sqsupseteq B_c)$**

2.1 Incompatibility

It is also possible to talk about incompatibility between two type descriptors. When we say that type A is incompatible with type B, what we mean is that there exists at least one characteristic in the type descriptor for A, which is incompatible with a characteristic in the type descriptor for B. (Once again it is not necessary to pinpoint what the characteristic is that makes A incompatible with B, though of course that would be extra usable information. It is enough to state that the incompatibility exists). For example if one type has the characteristic 'weight 5kg.' and another the characteristic 'weight 10kg.', then these two type descriptors are incompatible with one another in that no object can belong to both these types simultaneously.

The formal definitions of incompatibility are as follows:

Definition: There is a relation $\mathcal{I} \subseteq \mathbb{C} \times \mathbb{C}$ such that $\mathcal{I}(c,c')$ holds iff the characteristic c is incompatible with the characteristic c'.

We observe that $\mathcal{I}$ is irreflexive and intransitive but symmetric, i.e. the following is satisfied:

$$\forall c,c' \in \mathbb{C}: \mathcal{I}(c,c') \rightarrow \mathcal{I}(c',c).$$

Definition: The descriptor A is *incompatible,* written $\mathcal{I}(A)$, iff $\exists c,c' \in A: \mathcal{I}(c,c')$

It is assumed that no type descriptor or object descriptor is in itself incompatible.

We now define the notion of a complement of a type A, which intuitively consistes of the union of types whose characteristics are incompatible with A. This complement is written NOT(A).

Definition: NOT(A) is defined to be a set according to the following equation:

$$NOT(A) = \{B \mid \mathcal{I}(A \sqcup B) \wedge \neg \mathcal{I}(A) \wedge \neg \mathcal{I}(B)\}$$

Note that if A is inconsistent then NOT(A) is the empty set.

Proposition 1: $A \sqsubseteq B \wedge \mathcal{I}(A) \rightarrow \mathcal{I}(B)$. That is to say incompatibility in a descriptor is necessarily inherited by all more specific descriptors.

Proof:

$\sqsubseteq$ B implies $A \subseteq B$.
$\mathcal{I}(A)$ implies $\exists c,c' \in A: \mathcal{I}(c,c')$.
Since $A \subseteq B$, then $\forall c \in A: c \in B$, which implies
$\exists c,c' \in B: \mathcal{I}(c,c')$, which implies $\mathcal{I}(B)$. ¶

Proposition 2: $\mathcal{I}(A \sqcup B) \wedge (C \sqsupseteq A) \wedge \neg\mathcal{I}(C) \wedge \neg\mathcal{I}(B) \rightarrow C \in NOT(B)$. If a descriptor A is incompatible with a consistent descriptor B, then all consistent descriptors more specific than A are NOT(B).

Proof:

$C \sqsupseteq A$ implies $(A \sqcup B) \sqsubseteq (C \sqcup B)$, which together with $\mathcal{I}(A \sqcup B)$ implies $\mathcal{I}(C \sqcup B)$ (proposition 1)
which together with $\neg\mathcal{I}(C)$, $\neg\mathcal{I}(B)$ implies $C \in NOT(B)$. ¶

The incompatibility relation can of course be between any combination of core and default type pairs, giving similar nuances in negative statements as in positive. Thus we can say:

* **A's are never B's** $\mathcal{I}(A_c \sqcup B_c)$
* **Typical A's are never B's** $\mathcal{I}(A_d \sqcup B_c)$
* **A's are never typical B's** $\mathcal{I}(A_c \sqcup B_d)$
* **Typical A's are not typical B's.** $\mathcal{I}(A_d \sqcup B_d)$

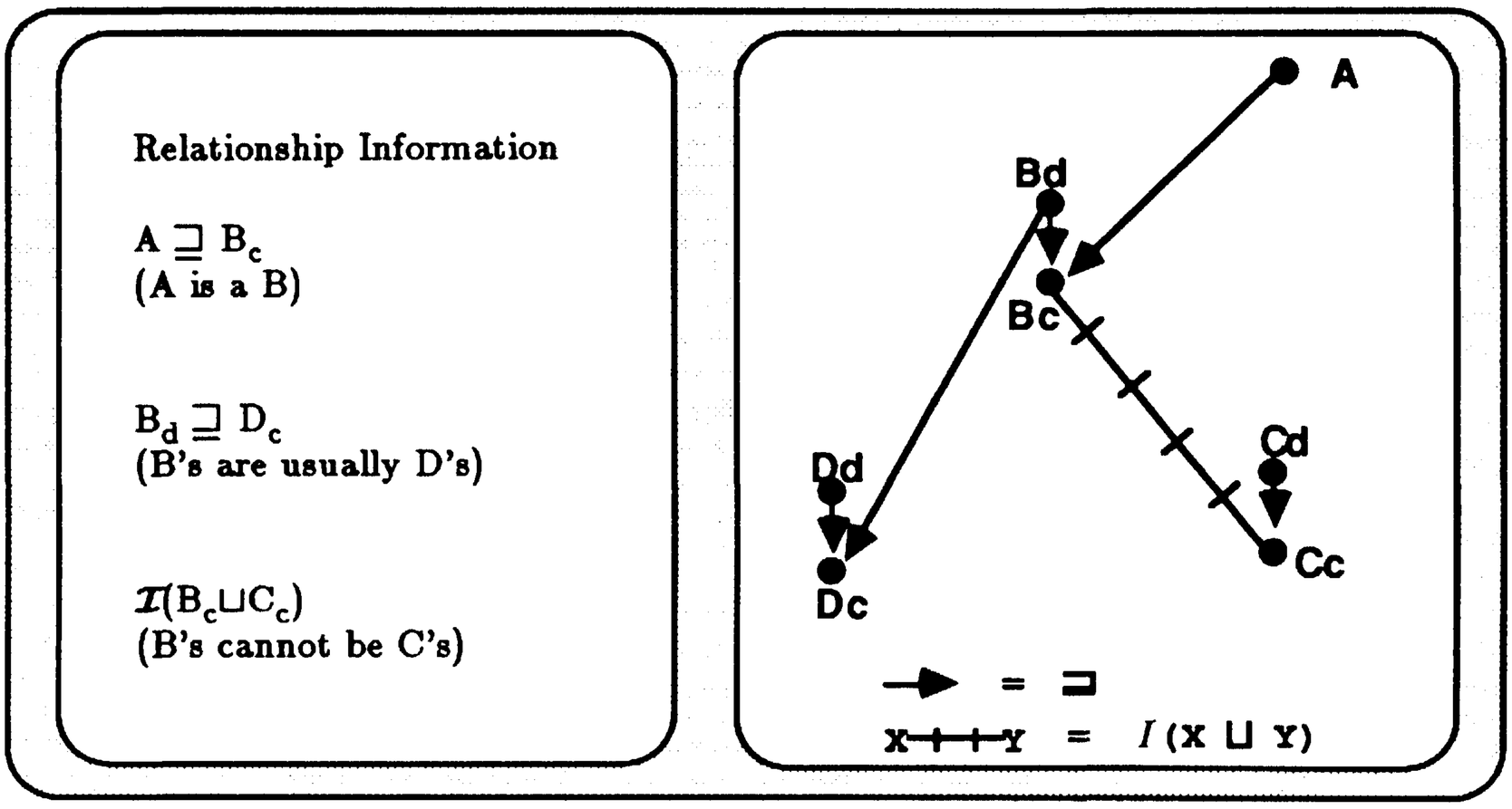

Fig. 1

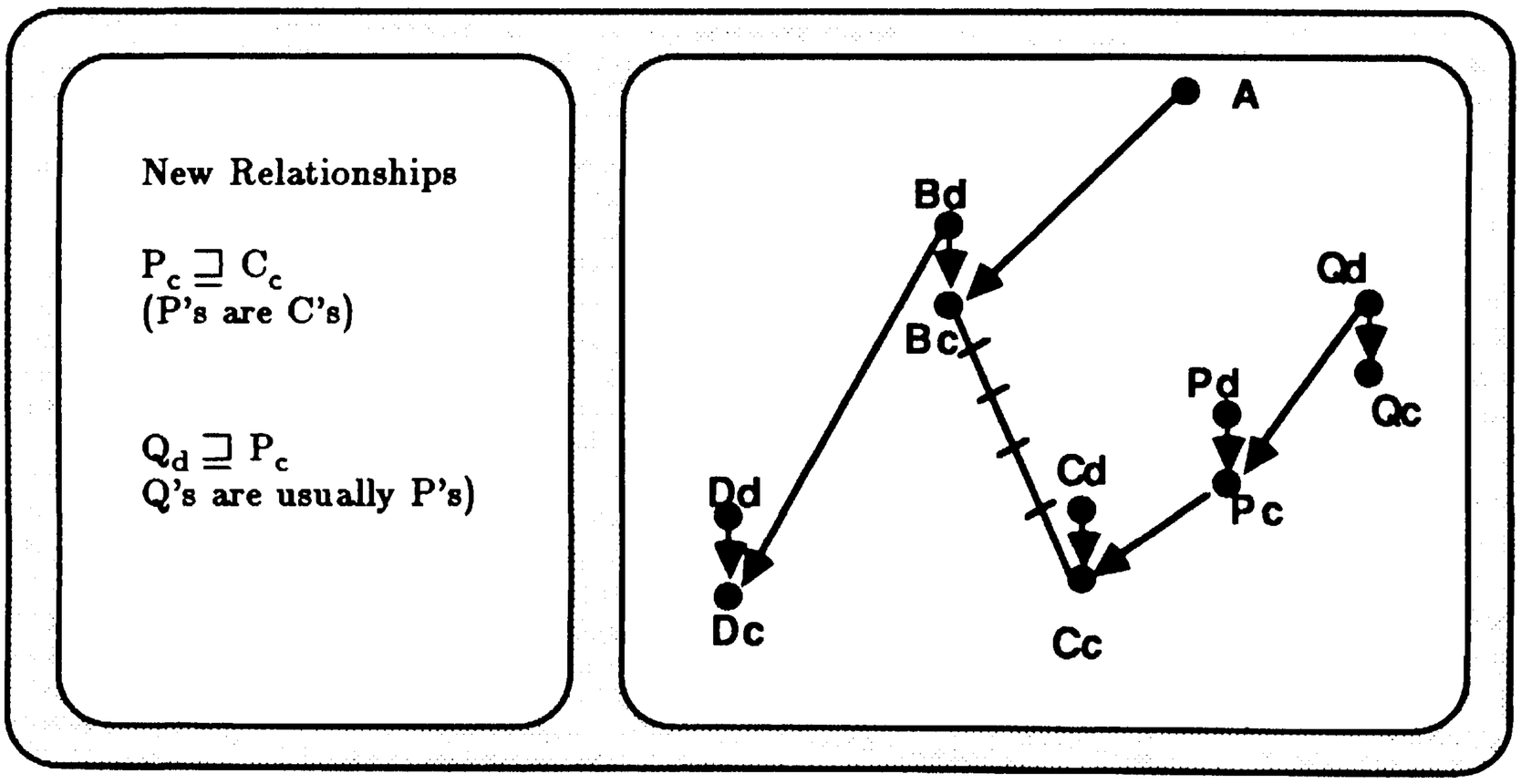

Fig. 2

The positive and negative relationships described here then make it possible to draw a diagram such as fig. 1, where we can follow paths through the graph to collect information. By defining a *default assumption* operation E which takes us from a core point to its corresponding default we can come to a new point in the graph, enabling us to collect further information. These jumps upward in information content, are essentially the reasoning step 'we know that we have an A, in the absence of information to the contrary, let us assume that we have a typical A.'

Definition: A default assumption, written $\mathcal{E}(T)$, is the step which allows us to add the relation 'object $\sqsupseteq$ T_d' to our set of relations.

We also allow $\mathcal{E}(T)$ to add in a relationship 'objec $\sqsupseteq$ T_d-δ', where T_d-δ is $\sqsupseteq$ T_c but has some of the T_d information removed (in order to deal with partial defaults). For further explanation of partial defaults see [Padgham 89].

Thus from fig. 1 we can say with certainty that A is a B, and is not a C. After making the default assumption that we have a typical B, we can also add the information that A is a D.

3. Negative Reasoning

If we now add extra information to fig 1, to obtain the diagram shown in fig. 2, what more can we say about A? Our claim is that we can say that if we accept the previous conclusions about A, then A is also definitely not a P, and probably not a Q. The reasoning that A is definitely not a P, we refer to as strict negative reasoning because given that we believe A is not a C, the extension to believing that A is also not a P does not require any default assumptions. The reasoning that A is also probably not a Q relies on the assumption that if A is definitely not a typical Q, then A is probably not a Q at all. Thus we call this default negative reasoning. More formal justification of the reasoning used to obtain these extra negative conclusions is given in the following two subsections.

3.1 Strict Negative Reasoning

We define *strict negative reasoning* as the reasoning which takes a negative conclusion regarding the incompatibility of the object being reasoned about with some type, and deduces all logically necessary negative conclusions based on this incompatibility. For example, if we have $\mathcal{I}$(object $\sqcup$ C) (i.e. we believe NOT(C)) and we know that P $\sqsupseteq$ C, then we are logically bound to believe NOT(P).

The logical necessity of the conclusions obtained by strict negative reasoning relies, of course, on acceptance of the basic tenets of the model, namely that:

* types can be defined (theoretically) by their core and default type descriptor each of which is a set of characteristics.
* these sets of characteristics can be compared with each other according to the more information relation $\sqsupseteq$
* incompat ability between types A and B can be explained as incomp at ability between at least one characteristic of type A and one characteristic of type B.

Given that we accept the model, proof for the correctness of strict negative reasoning is as follows:

Proposition 3: If X,Y,W,Z are type descriptors such that Y $\sqsubseteq$ X, W $\sqsubseteq$ Z. and Y $\sqcup$ W is incompatible, then X € NOT(Z).

Proof:

$X \sqsupseteq Y$ implies $X = X \sqcup Y$, and
$Z \sqsupseteq W$ implies $Z = Z \sqcup W$.
Therefore the following is satisfied:
$$X \sqcup Z = (X \sqcup Y) \sqcup (Z \sqcup W)$$

Since $\sqcup$ is commutative and associative we obtain $X \sqcup Z = (Y \sqcup W) \sqcup (X \sqcup Z)$ which implies $(X \sqcup Z) \sqsupseteq (Y \sqcup W)$.

Since Y $\sqcup$ W is incompatible and $\sqsupseteq$ has been shown to preserve incompatibility, it follows that X $\sqcup$ Z is incompatible which implies X € NOT(Z). ¶

We note that in the situation where we have C $\sqsubseteq$ P_d, we are not logically bound to add in NOT(P), as we do not have a definite incompatibility with P_c. I.e. C is-not-a P means only C E NOT(P_c).

If there have been default assumptions made on the way to the conclusion on which negative reasoning is based, then the negative conclusions which follow are of course default conclusions rather than certain conclusions. However this reasoning is referred to as strict (or monotonic), because if the initial incompatibility is believed then we are logically bound to also believe the consequences of strict negative reasoning based on the incompatibility. To not do so results in a set of conclusions which is either incomplete or inconsistent.

3.2 Default Negative Reasoning

Default negative reasoning is the step that can be intuitively described as 'if we know that what we have does not fit the description for a typical X, then let us assume that it is not an X.' What we do formally is to define an operation N which allows us given $\mathcal{I}(A \sqcup X_d)$ to assume $\mathcal{I}(A \sqcup X_c)$, thus adding the information A € NOT(X). Following such an operation we can then continue our strict negative reasoning.

Definition: A negative default assumption, written N(T) is the step which allows us to, given the conclusion $\mathcal{I}(A \sqcup T_d)$, add the relation $\mathcal{I}(A \sqcup T_c)$.

That is if we believe that the object about which we are reasoning is an element of NOT(T_d) then we assume that it is also an element of NOT(T_c).

This default negative reasoning is not provably correct in the way that the strict negative reasoning is, just because it is default reasoning, and as such requires an assumption. However the nature of the assumption required is essentially symmetrical to that required for positive default reasoning. It also seems intuitively reasonable.

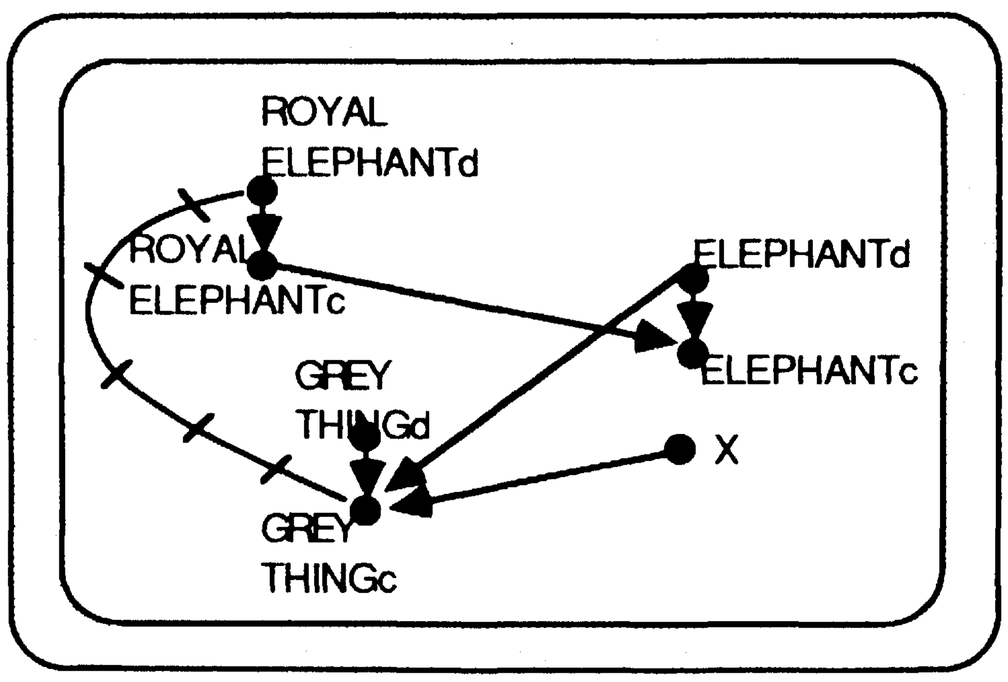

Fig. 3

Using the example in fig. 3 default negative reasoning gives the effect that if we have an object X which is a Grey Thing, then we know that $\mathcal{I}(X \sqcup \text{RoyalElephant}_d)$ (because X $\sqsupseteq$ Grey Thing and I(GreyThing $\sqcup$ RoyalElephant_d)) N(RoyalElephant) then allows us to add the relation $\mathcal{I}(X \sqcup \text{RoyalElephant}_c)$, thus giving the conclusion that X is not a Royal Elephant. This seems intuitively reasonable, and in keeping with human reasoning.

3.3 Preference for Specificity

When doing positive reasoning one can obtain conclusions that conflict with one another, by virtue of the fact that positive reasoning can result in a final negative conclusion. When such conflicts occur, the decision as to what should be preferred is based on a notion of specificity, preferring conclusions which rely on default assumptions at the most specific level possible.[Padgham 88]

In negative reasoning it is not possible to arrive at conclusions which conflict with other conclusions derived via negative reasoning. This is because the negative reasoning methods can only lead to negative conclusions, making it impossible to obtain both X and NOT X in the negative reasoning phase. Consequently conclusions based on negative reasoning methods do not have to be ordered according to the specificity of the default assumption on which they are based. However conclusions obtained during negative reasoning can conflict with conclusions obtained during positive reasoning.

Conclusions obtained by strict negative reasoning from a conclusion (e.g. NOT(X)) that was obtained by deduction using *S* should clearly be resolved using the specificity preference. The conclusions of the strict negative reasoning depend on the same default assumption as the conclusion NOT(X) itself depends on. This must then be compared for specificity with the default assumption on which the conflicting conclusion depends.

For those negative conclusions which depend on a negative default assumption, it is less clear how they relate to conflicting conclusions based on a positive default assumption. It seems difficult to use specificity in an intuitive and well defined way in that it is not entirely clear how the specificity of NOT A's should be compared to the specificity of B's. An initial intuition is that positive default assumptions may always be preferred over negative default assumptions.

4. Example Using Negative Reasoning

Let us look now at a concrete example where the kind of negative reasoning described could be used. Let us suppose that we have a decision support system for general medical practitioners which includes, amongst other things, a knowledge base of various types of drugs, organised according to our lattice based model.

One of the things which doctors must be very

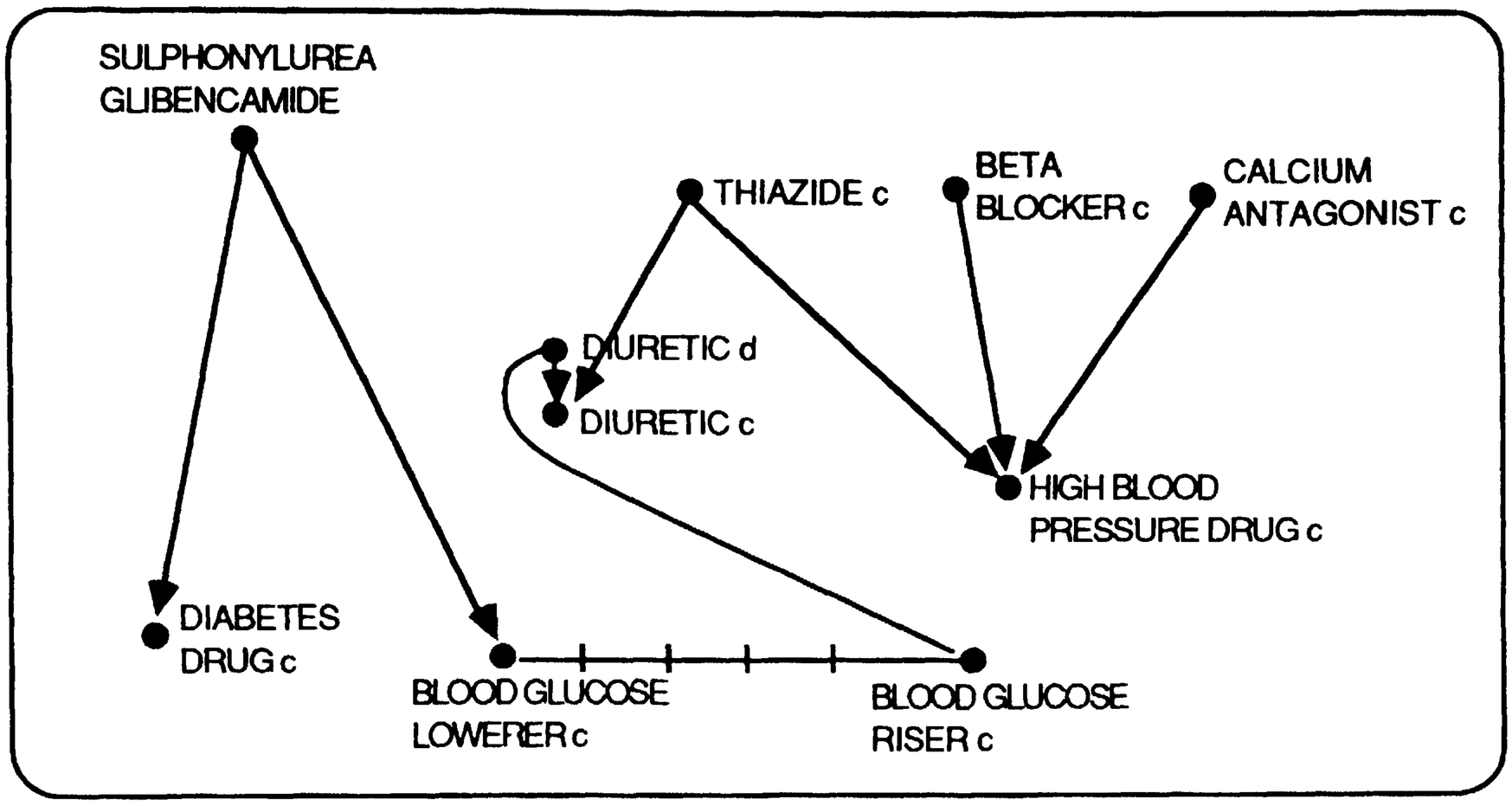

Fig. 4

concerned about when prescribing drugs is to be aware of contra-indications for giving a particular drug. Contra-indications are often other drugs that the patient may be taking (or that the doctor is about to prescribe). Thus the doctor must consider the entire drug group that the patient will take and ascertain that it is compatible.

Let us imagine now that a patient comes to the doctor and is diagnosed as having high blood pressure which needs to be treated by medication. The patient is also a diabetic and is currently taking medication for diabetes. The doctor must ensure that the blood pressure medication he prescribes does not conflict with the diabetes medication.

A part of the knowledge base of drugs is shown in fig. 4. If we know that the patient is currently taking Sulphonylurea Glibencamide for diabetes, we can use the inheritance reasoning strategies described in this paper to ascertain that we should not prescribe any of the thiazides for treating the patient's high blood pressure, as this will lead to an incompatibility regarding what is happening with the blood glucose level.

If we refer to the group of drugs which the patient is going to take as D, then we can use ordinary default /inheritance reasoning (following links and using the *€* operation), to collect the information *Sulphonylurea Glibencamide, Diabetic treatment drug, Blood glucose lowerer, NOTfblood glucose raiser).* We can then continue further using negative reasoning, starting from the negative conclusion *NOT(Blood glucose raiser).* By following the links in a backward direction, and using the default operation *J\f* to go from default to core of diuretics, we collect the additional information *NOTfdiuretic), NOT (thiazides).*

Given that thiazides are in fact one of the groups of drugs which are high blood pressure drugs, it is very useful to be able to obtain the information that a consistent group of drugs D, which contains Sulphonylurea Glibencamide may not contain thiazides. In this case it is easily possible to choose one of the other groups of high blood pressure drugs (Calcium antagonists or Beta blockers), which do not lead to any incompatibility. If there were no alternative drug groups then there are a number of other reasonable possibilities, e.g. try to find a thiazide that is an atypical diuretic and is not a blood glucose raiser, or alter the drug used for diabetes treatment. However these questions are outside the scope of the present paper.

The model is only relevant for cases where there are no complex interactions between the two drugs which cannot be expressed as the union of their combined characteristics. Also it may be that the inconsistency detected is not always relevant. For instance if 'blood glucose lowerer' had been an unimportant side effect of the diabetes medication, rather than the desired effect, then it would not have mattered that this is blocked by some other medication. Reasoning beyond the detection of the inconsistency can either be done by the doctor or by some other reasoning module, but is not part of the mechanisms described here.

5. Conclusions

The lattice based model of typing schemas presented previously [Padgham 88] extends naturally to include reasoning about what an object is NOT. A subset of the results of this extension are similar to results reported by Horty and Thomason [88][1]. However this model provides a proof of correctness for the results, which are obtained simply by definition within their model. The extra results which come from the default negative reasoning are also considered to be important.

The results of strict negative reasoning are available in any model based reasoning system based on classical logic (e.g. circumscription [McCarthy 1986], default logic [Reiter 1980] etc.), as following backwards along the links from a negative conclusion amounts simply to taking the contrapositive form of an implication. Those approaches such as circumscription where the logic is classical, and the default behaviour comes from minimising some sort of abnormality predicate will also obtain the conclusions obtained by our N operator. However such systems cannot distinguish between conclusions obtained by contraposing (or use of modus tolens) and conclusions obtained by other mechanisms equivalent in our system to use of the *E* operator. This lack of distinction can lead to many extensions with no structural mechanism available to choose between them. We believe that having A/' as a separate operator, allows both the power of drawing extra conclusions as well as the ability to discriminate and make distinctions. This ability to discriminate is needed when one has many possible conclusions but wishes to prefer some conclusion sets over others.

The model provides for clean and simple determination of such things as correct pre-emption behaviour and which graphs are inconsistent in the information they contain, and can therefore not be expected to lead to correct/meaningful results. More work is needed in the area of default negative reasoning, to determine what priority conclusions obtained from this reasoning should have with respect to positive conclusions. These issues have not been explored in the literature because (at least as far as this author is aware) there have not been systems which are able to both use contraposition and distinguish it from other reasoning methods. One possibility is to compare the specificity of the level at which the N operation is used, with the specificity of the level at which *S* is used to obtain the competing positive conclusion. However our intuition is that we should instead prefer all positive conclusions over negative conclusions obtained using default negative reasoning. It may be however, that preference is determined by what one wishes to use the reasoner for. It is a distinct advantage of the presented model that such changes are simple and clean to make in the algorithms which manipulate the model, and do not require a change within the model itself.

1 Based on personal communication (Horty, Feb. '89) the beliefs supported by strict negative reasoning appear to be essentially equivalent to those obtained by Horty's definition of a skeptical reasoner using mixed link types. We do not have any proof of equivalence.

Acknowledgements

I thank Ralph Ronnquist for help with the formal expressions, and Toomas Timpka for help with the medical example.

References

ETH87

Etherington, D. W. Formalizing Nonmonotonic Reasoning Systems. *Artificial Intelligence,* vol. 31, 1987, pp. 41-85.

FAH79

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

HOR88

Horty, J. F., Thomason, R. H. Mixing Strict and Defeasible Inheritance. *Proceedings of AAAI '88* St Paul, Minnesota, August 20-26, 1988, vol 2, pp. es427-432

PAD88

Padgham, L. A Model and Representation for Type Information and its Use in Reasoning with Defaults. *Proceedings of AAAI '88* St Paul, Minnesota, August 20-26, 1988, vol 2, pp.409-414

REI80

Reiter, R. A Logic for Default reasoning, *Artificial Intelligence 13,* North-Holland, 81-132

SAN86

Sandewall, E. Non-monotonic Inference Rules for Multiple Inheritance with Exceptions. *Proceedings of the IEEE,* vol 74, 1986, pp. 1345-1353.

TOU86

Touretzky, D.S. *The Mathematics of Inheritance Systems.* Morgan Kaufmann Publishers, Los Altos, CA, 1986.

MCC86

Applications of Circumscription to Formalizing Common Sense Knowledge, *Artificial Intelligence Journal,* vol 28, pp. 89-116