THE LOGIC OF INHERITANCE IN FRAME SYSTEMS

Gerhard Brewka
German Research Institute for Mathematics and Data Processing
Postfach 12 40
D 5205 Sankt Augustin
fed. Rep. of Germany

ABSTRACT

This paper shows how the semantics of frames with exceptions can be described logically. We define a simple (purely declarative) frame language allowing for multiple inheritance and meta classes (i.e. the instances of a class may be classes themselves). Expressions of this language are translated into first order formulas. Circumscription of a certain predicate in the resulting theory yields the desired semantics. Our approach allows the intuition that subclasses should override superclasses to be represented in a very natural way.

I, INTRODUCTION

Inheritance systems have a long tradition in AI. They allow the description of hierarchies of objects and classes (we use the term hierarchy in the sense of an acyclic network throughout the paper) and the inheritance of properties in such hierarchies. There are two main types of inheritance systems: those which admit exceptions to inheritance and those who do not. KL-ONE [Brachman/Schmolze 85] and OMEGA [Attardi/Simi 81] are examples of the second type. Systems with exceptions are for instance FRL [Roberts/Goldstein 77], Flavors [Symbolics 85] and the BABYLON-Frame-System [di Primio/Brewka 85].

A common way of defining a semantics for a formalism F_1 is to translate it into another formalism *TZ* for which a well defined semantics exists (this amounts to stating that FI is a notational variant of (a subset of) F2). In the knowledge representation field first order logic is a good candidate for F2 since it is well understood and has an appealing semantics.

It is not too difficult to describe the semantics of inheritance systems without exceptions in first order logic, see for instance [Hayes 79], [Hayes/Hendrix 81], where frames are interpreted as unary and slots as binary predicates.

Inheritance with exceptions is much more difficult since exceptions introduce nonmonotonicity. Etherington and Reiter use default logic [Reiter 80] to describe the semantics of NETL-like inheritance networks [Etherington/Reiter 83][Etherington 87]. The problem with their approach is that the intuition underlying all inheritance systems, namely that subclasses should override superclasses (we will call this intuition the *specialization principle),* is not present in their formalization. They require exceptions to be explicit (as exception links) in the network.

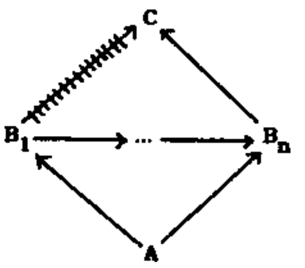
Touretzky tries to capture the specialization principle in his inferential distance ordering [Touretzky 84] [Touretzky 86]. But 1 think he is not right in stating that "inferential distance is a partial ordering on defaults that implements this intuition" ([Touretzky 84],p.325). The reason is very simple. The inheritance system he uses cannot represent the notion of subclass. It only uses defeasible links with the intuitive meaning *A's are typically B's.* But if we have long chains as

A_1*'s are typically* A_2*'s*
A_2*'s are typically* A_3*'s*
...
A_{n-1}*'s are typically* A_n*'s*

it may even be the case that A_1. and A are disjoint. But how can the intuition that subclasses should override superclasses be captured if the notion of subclass cannot even be represented? Touretzky simply redefines the meaning of *subclass.* He states: " ... A is a subclass of B iff there is an inheritance path from A to B."([Touretzky 84],p.322). But changing the definition of subclass is certainly not a solution.

As Sandewall [Sandewall 86] has shown, there are cases where Touretzky's approach leads to counterintuitive results. We will discuss an example in section VI.

Sandewall argues for defining a partial semantics for inheritance systems by a collection of structure types since this "semantics by examples" is the best we have up to now. He, also, does not distinguish between strict and defeasible links . This can lead to situations where we expect different results from the same structure type. Let's discuss a slight generalization of his type IB structures which can be represented as follows (IS-A links are represented by , NOT-IS-A links by



With n = 2 and the interpretation A = Clyde, B_1 = RoyalElephant, B_2 = Elephant, C = Gray (this is exactly Sandewall's example) we expect, of course, Clyde not to be Gray. But what if we have n = 4 and A = Peter, B_1 = Student, B_2 = Adult, B_3 = Married, B_4 = HasTwoChildren, C = OlderThan28. I think there is no reason to prefer Peter being not OlderThan28.

We try to show in this paper that at least for special types of inheritance systems the situation is not that bad. We will concentrate on frame systems (class/property inheritance systems in Touretzky's terminology [Touretzky 86]). They allow to represent hierarchies with strict subclass relations as well as properties (slots with slot values) that members of classes (frames) typically possess. On one hand the expressiveness of these systems is stronger than that of inheritance systems which only admit *IS-A* and *NOT-IS-A* links since defeasible and undefeasible links can be represented. On the other hand the expressiveness is much more restricted since chains of defeasible inferences are not allowed. Fortunately, we do not need Touretzky's formal apparatus and his predicate lattices to describe the semantics of such a frame system.

In principle, second order logic would be a good formalism to express the specialization principle in a very intuitive way. If frames are interpreted as predicates, then the fact that one frame specializes another can easily be represented by a second order predicate. But, of course, second order logic introduces many difficulties, especially since we need a nonmonotonic logical formalism.

Fortunately, there is a well-known technique (sometimes called "reiflcation") which allows us to remain totally in first order logic. Instead of writing MAN(Peter) for expressing the fact that Peter is an instance of the frame Man, we introduce a predicate IS and a constant Man and express the mentioned fact as IS(Peter.Man). If a slot Age of Peter has the value 28. then we write HOLDS(Age,Peter,25) instead of AGE(Peter,28) The use of constants instead of predicate symbols allows to reason about properties of frames in first order logic.

We will introduce a three-place predicate EXCEPTIONAL whose extension will be minimized. We use McCarthy's circumscription technique [McCarthy 84] for that purpose. Circumscription is a very general formalization of nonmonotonic reasoning, many even think the most promising one (e.g. Raymond Reiter stated that in his invited talk during AAAI-86). The advantages of circumscription are

1) it is quite close to standard logic,

2) it has a very appealing semantics defined in terms of minimal models.

Various different forms of circumscription have been defined during the last years. We will need a form known as *variable circumscription* [McCarthy 84]. Variable circumscription allows predicates to vary during the minimization. It will turn out that the logical formulas we need to represent frame and instance definitions belong to the class of universal formulas. This is fine since for universal formulas variable circumscription cannot lead to inconsistencies [Lifschitz 86].

The rest of this paper is organized as follows. In section II we define a simple frame language which allows us to define a class/instance hierarchy with multiple inheritance and meta classes (i.e. classes whose instances are classes themselves). In section III we show how the definitions of this language can be translated into a set of first order formulas in which the predicate EXCEPTIONAL is circumscribed. Section IV gives a short example, and in section V we discuss ambiguities in frame systems.

Throughout the paper we use a logic with equality and make the unique names assumption, i.e. different symbols denote different objects.

II. THE FRAME LANGUAGE

The frame language we will use in this paper is very similar to the BABYLON frame language [di Primio/Brewka 85], but extends it somewhat since we want to allow frames to be instances of (meta) frames. Frames are defined as follows:

```
(defframe frame-name
  {(supers superframe_1 ... superframe_l)}
  {(slots
     (slot_1 value_1)
     ...
     (slot_k value_k))})
```

{}-brackets indicate an optional part of a definition. Instances of frames can be defined in the following way

```
(definstance instance-name of frame-name
  {with
    slot_1 = value_1
    ...
    slot_h = value_h})
```

For the sake of simplicity we do not allow instances to be instances of different frames here. This is not a strong restriction, because we always can define one frame having the different frames as supers and instantiate that frame.

Since we want to allow instances to be classes themselves, we extend the usual frame definitions somewhat and allow also definitions of the form

```
(defframe frame-name_1
  {(instance-of frame-name_2
    {with
      slot_m = value_m
      ...
      slot_n = value_n})}
  {(supers superframe_1 ... superframe_q)}
  {(slots
     (slot_k value_k)
     ...
     (slot_l value_l))})
```

In this definition *slot*$_m$ ... *slot*$_n$ are attributes of the frame itself, *slot*$_k$ ... *slot*$_l$ attributes of the instances of the frame (in our language the difference betweeen instances and frames is less important than in other frame systems, so we could have used a common construct, e.g. defobject for all definitions. We wanted, however, to retain some similarity with other well known languages and chose the more conventional syntax, therefore).

Let us assume that frames must be defined before they are mentioned in definitions as supers or as frames to be instantiated. Then our language allows to define directed acyclic graphs in which superclass and instance links may appear anywhere, but - as mentioned above - at most one instance link may go out from one node.

We allow multiple inheritance in our frame language. This means that a frame is allowed to have more than one (direct) superframe. In this case an inheritance strategy must specify from which frame an object inherits a property. There are different strategies implemented in existing systems. Mostly the (direct or indirect) superframes of a frame are linearized in some way. The Flavors system [Symbolics 85], for instance, uses such a linearization (until recently even superclasses could override their own subclasses in certain cases, this has been remedied in the newest Flavors version). This is not the best idea, however, since sometimes any linearization yields unintuitive results. An old example is: quakers are pacifists, republicans are not pacifists, Nixon is a quaker and a republican. How about his pacifism? The available information is ambiguous. Our intuition in this case is to remain agnostic. We have evidence for and against his pacifism, but nothing allows to prefer one of the contradictory conclusions. So we simply don't know and our system should not infer anything.

In the introduction we mentioned the specialization principle: subclasses should override superclasses. Here we have the other side of the coin: only subclasses should override superclasses. This is also Touretzky's view [Touretzky 86].

In section V we will show that circumscription easily handles ambiguities the way we want them to be handled.

III THE MEANING OF THE FRAME LANGUAGE

We now describe how definitions of our frame language can be translated into a set of second order formulas. We introduce the predicate EXCEPTIONAL which essentially is a three-place variant of McCarthy's ABnormal predicate. The idea is: if $Frame_1$ has a slot Slot| with value $Value_1$ this will be represented as

Forall x.
  IS(x,Frame.) ft - EXCEPTIONAL(x.Slot. $Frame_1$)
  ->
  HOLDS(Slot 1 .x.Value 1*)*

Intuitively EXCEPTIONAL(x,Slot1 Frame $_1$ can be read as "x does not inherit information about attribute $Slot_1$ from frame Frame.".

Sometimes slots are interpreted as functions, not as general relations. We prefer the second approach here since it makes multiple values for slots possible. We don't deal with multiple values in this paper, however.

Independently from the definitions to be translated we need the following four formulas:

1) *[meaning of SPECIALIZES]*

Forall p.q.
  SPECIALIZES (p.q)
  ->
  (Forall x. IS(x.p) -> IS(x,q))

If a class specializes another class then all members of the class are also members of the other class.

Z) [transitivity of SPECIALIZES]

Forall x,y,z.
  SPECIALIZES(x,y) * SPECIALIZES(y.z)
  ->
  SPECIALIZES(x,x)

Specialization is transitive.

3) *[meaning of HAS-SLOT]*

Forall frame.slot.value.
  HAS-SLOT(frame,Blot, value)
  ->
  (Forall x.
    IS(x.frame) & - EXCEPT10NAL(x.slot.frame)
    ->
    HOLDS(slot,x,value))

We introduce the predicate HAS-SLOT here as a matter of convenience. It makes the rest of the translation more readable.

4) *[specialization formula]*

Forall x. frame.. $frame_2$. value,. $value_2$. slot.
  IS (x.frame.) 4c
  HAS-SLOT(frame..slot.value.) &
  SPECIALIZES (frame. ,f rame2) &
  HAS-SLOT ($frame_2$.srot,$value_2$) &
  - value. = $value_2$

  EXCEPTIONAL(x,alot.frame2)

Intuitively: x is exceptional with respect to a slot of a frame if x is an instance of a more special frame, for which different information regarding the slot is available. This is the formula representing the specialization principle.

Now we can easily map frame and instance definitions into a set of logical formulas.

The definition of a frame

(defframe *my-frame*
  (supers *superframe* $_J$. ... *superframe.)*
  (slots $\quad$ *
    (slot $_1$ value $_1$)

    (slot1value1)))

is translated into the following formulas

SPECIALIZES(my-frameup*r/romc $_;$)

SPECIAUZES(my-/rams .*superframe $_k$)*

HAS-SLOT(my-/ram« *Mot $_1$ .value $_1$*)

HAS-SLOT(my-/ram«$_t$sio*$_n$,voiu«$_n$)

The definition of an instance

```
(definstance my-instance of my-frame with
   slot_1 = value_1
   ...
   slot_h = value_h)
```

yields the following formulas:

```
IS(my-instance,my-frame)

HOLDS(slot_1,my-instance,value_1)
...

HOLDS(slot_h,my-instance,value_h)

Forall frame, value.
  IS(my-instance,frame) &
  HAS-SLOT(frame,slot_1,value) &
  ~ value = value_1
  ->
  EXCEPTIONAL(my-instance,slot_1,frame)
 ...

Forall frame, value.
  IS(my-instance,frame) &
  HAS-SLOT(frame,slot_h,value) &
  ~ value = value_h
  ->
  EXCEPTIONAL(my-instance,slot_h,frame)
```

The last implications guarantee cancellation of inheritance for slots initialized in the instance definition.

Definitions of the form

```
  (defframe frame-name_1
   (instance-of frame-name_2
     with
      slot_m = value_m
      ...
      slot_n = value_n)
   (supers superframe_1 ... superframe_q)
   (slots
     (slot_k value_k)
     ...
     (slot_l value_l)))
```

have the same translation as the two seperate definitions

```
(defframe frame-name_1
  (supers superframe_1 ... superframe_q)
  (slots (slot_k value_k)
          ...
          (slot_l value_l)))
```

and

```
(definstance frame-name_1 of frame-name_2 with
   slot_m = value_m
   ...
   slot_n = value_n)
```

The meaning of a set of frame and instance definitions now is the variable circumscription of **EXCEPTIONAL** in the set of formulas described above. We allow all predicates to vary during the minimization, cf. [McCarthy 84].

IV. EXAMPLE

We use the following definitions

```
(defframe Car
  (slots
    (Wheels 4)
    (Seats 5)))

(defframe Sportscar
  (supers Car)
  (slots
    (Seats 2)
    (Cylinders 6)))

(definstance Speedy of Sportscar with
  Cylinders = 8)
```

then, besides formulas 1) ... 4) as defined in section III, we get the following formulas:

```
HAS-SLOT (Car,Wheels,4)
HAS-SLOT (Car,Seats,5)
SPECIALIZES (Sportscar,Car)
HAS-SLOT (Sportscar,Seats,2)
HAS-SLOT (Sportscar,Cylinders,6)
IS(Speedy,Sportscar)
HOLDS(Cylinders,Speedy,8)
Forall frame, value.
  IS(Speedy,frame) &
  HAS-SLOT(frame,Cylinders,value) &
   ~ value = 8
  ->
  EXCEPTIONAL(Speedy,Cylinders,frame)
```

Circumscribing **EXCEPTIONAL** in this theory (allowing all predicates to vary) yields

```
Forall x,y,z.
  EXCEPTIONAL(x,y,z)
  <->
  (x = Speedy & y = Seats & z = Car)
  v
  (x = Speedy & y = Cylinders & z = Sportscar)
```

To derive interesting results we must assume, of course, that different names denote different objects (e.g. that cylinders and seats are not the same). But under the unique names assumption we get exactly the results we expect:

HOLDS(Wheels,Speedy,4) is derivable since **Speedy** is a **Car** and not **EXCEPTIONAL** with respect to **Wheels** and **Car**.

HOLDS(Seats,Speedy,2) can be derived since **Speedy** is not **EXCEPTIONAL** with respect to **Seats** and **Sportscar**. **EXCEPTIONAL(Speedy,Seats,Car)** holds, however, so there are no inconsistencies.

HOLDS(Cylinders,Speedy,8) was asserted directly in the translation; there are no inconsistencies since **EXCEPTIONAL(Speedy,Cylinders,Sportscar)** holds.

V. AMBIGUITIES

Up to now our formalization still admits different values for a slot inherited from different paths in the frame network. Usually such situations are interpreted as ambiguities and, as discussed in section II, the best thing to do in case of an ambiguity is to remain agnostic. It is very easy to achieve the desired behavior in our formalization, since circumscription does most of the job automatically. We simply have to add

```
Forall slot.x,valuel.value2.
  HOLDS(slot,x,Taluel) &
  H0LDS(slot.x.value2)
  ->
  value1 = value2
```

With this additional formula we forbid slots having different values (of course, we could easily extend our frame language and distinguish between different types of slots, e.g. multi-valued and single-valued slots. In that case we should restrict the above implication to single-valued slots by introducing an additional condition).

Circumscription is a correct realization of minimal entailment. In our formalization ambiguities correspond to different minimal models. Since circumscription only allows to derive what is true in all minimal models, we get exactly what we want.

Let us assume we have a frame QUAKER with slot POLITICAL-VIEW and value PACIFISTIC and a frame REPUBLICAN with the same slot and value NON-PACIFISTIC. Let us also assume that none of the frames specializes the other. If NIXON is an instance of both frames, we have minimal models where H0LDS(P01JTICAL-VIEW. NIXON. PACIFISTIC) is true and HOLDS(POLITICAL-VIEW. NIXON. NON-PACIFISTIC) false But there also exist minimal models which make the first formula false and the second true. Circumscription only allows to derive the disjunction of both formulas but does not favour one of them.

VI. FUTURE WORK AND DISCUSSION

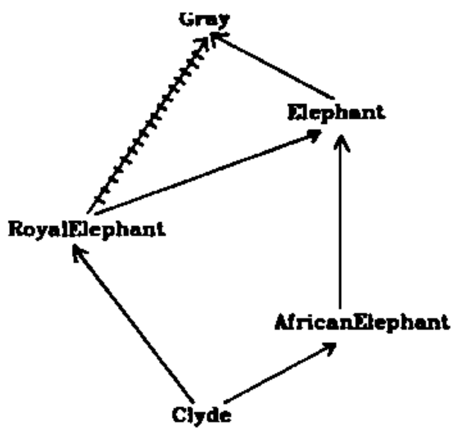
Many frame systems allow additional information to be associated with slots (often called facets). For the sake of simplicity we did not deal with facets in this paper, but it is straightforward to extend our approach accordingly and to represent them as well. Sometimes the facets are given a special meaning in frame systems. For instance, a facet POSSIBLE-VALUES of a slot could restrict the values that this slot may have. If. for instance, the POSSIBLE-VALUES facet of SLOT1 of FRAME1 has the value POSS-VAL-PRED, we can represent this as

```
Forall x. y.
  IS(x.Frame)   & HOLDS(Slot1 .x.y)

  POSS-VAL-PRED(y)
```

Of course, POSS-VAL-PRED has to be defined according to its intended meaning. In a more elaborate formalization, also values representing *unknown* or *undetermined* need special treatment (e.g. if they participate in ambiguities). But there are no theoretical difficulties.

Inheritance in general is very hard to deal with. We have shown, however, that for simpler classes of inheritance systems quite natural logical representations exist. Of course, it would be fine to see how our formalization deals with Sandewall's structure types [Sandewall 86]. Obviously, we cannot match the frame hierarchies definable in our frame language directly against Sandewall's examples since he admits only defeasible links (i.e. there is no guarantee that every royal elephant is an elephant) and we cannot represent chains of defeasible inferences. But for most of the examples an analogous representation in terms of frame hierarchies exists and we get the desired results. Let us discuss the example for which Touretzky's approach fails:



To represent this example in our language, we have to define Clyde as an instance of a frame RoyalAfricanElephant which has RoyalElephant and AfricanElephant as superframes. The last two frames each have the superframe Elephant. For RoyalElephant and Elephant a slot Color must be defined with value Non-Gray and Gray respectively. Now our formalization yields exactly what we expect, i.e. HOLDS(Color. Clyde, Non-Gray).

We have defined the semantics of a frame system with exceptions by means of circumscribing a certain predicate in a first order theory. Our approach formalizes the intuition that subclasses and only subclasses should override superclasses in a very natural way. Expressions of our frame language can be translated independently from the translation of other expressions. Moreover we don't need a complicated new mathematical apparatus. It should be mentioned, however, that most current implementations of frame systems (in fact all implementations I know of) are not correct with respect to the proposed semantics since they favour one inheritance path instead of remaining agnostic in case of ambiguities.

ACKNOWLEDGEMENTS

Thanks to all members of our research group, especially Thomas Christaller, Tom Gordon, Joachim Hertzberg, Franco di Primio and Karl Wittur. Tom also helped to make this paper as English as possible. Of course, the paper totally depends on the fundamental work of John McCarthy. Without his ideas no part of it could have been written.

REFERENCES

[Attardi/Simi 81]
Attardi, G., Simi, M., Consist9ncy and Completeness of Omega, a Logic for Knowledge Representation. Proc. IJCAI81, 1981,504-510

[Brachman/Schmolze 85]
Brachman. R.J.. Schmolze. J.G., An Overview of the Kir ONE Knowledge Representation System. Cognitive Science 9(2), 1985, 171-216

[Etherington 87]
Etherington. D.W., Formalising Nonmonotonic Reasoning Systems. Artificial Intelligence 31. 1987. 41-85

[Etherington/Reiter 83]
Etherington. D.W.. Reiter, R., On Inheritance Hisrarchies With Exceptions. Proc. AAA I-83, 1983. 104-108

[Hayes 79]
Hayes, P.J.. The Logic of Frames in: Brachman. R.J., Levesque. H.J. (Eds), Readings in Artificial Intelligence. Tioga Publishing. Palo Alto. 1979. 287-295

[Hayes/Hendrix 81]
Hayes, P.J.. Hendrix. G.G., A Logical View of Typos. SIGART Newsletter 74. 1981. 128-130

[Lifschitz 86]
Lifschitz, V., On the Satisfiability of Circumscription. Artificial Intelligence 28. 1986. 17-27

[McCarthy 84]
McCarthy. J.. Applications of Circumscription to Formalizing Common Sense Knowledge Proc. Non-Monotonic Reasoning Workshop. 1984. 295-327

[di Primio/Brewka 85]
di Primio. F.. Brewka. G.. BABYLON- Kernel Sy9t9m of an Integrated Environmont for Export System Development and Operation. Proc. 5th International Workshop Expert Systems and their Applications. Avignon. 1985. 573-583

[Reiter 80]
Reiter. R.. A Logic for Do fault Reasoning. Artificial Intelligence 25(1). 1980, 81-132

[Roberts/Goldstein 77]
Roberts. R.B.. Goldstein. IP.. The FRL Manual. MIT AI-Memo 409. 1977

[Sandewall 86]
Sandewall, E.. Nonmonotonic Inference Rules for Multiple Inheritance with Exceptions. Proc. IEEE, Vol. 74, No. 10. 1986. 1345-1353

[Symbolics 85]
Symbolics. Lisp Machine Manual. Vol. 3, Lisp Language, Cambridge. 1985

[Touretzky 84]
Touretzky, D.S.. Implicit Ordering of Defaults in InHoritanc9 Systems. Proc. AAAI-84, 1984. 322-325

[Touretzky 86]
Touretzky, D.S., The Mathematics of Inheritance Systems Research Notes in Artificial Intelligence, Pitman, London, 1986