Inheriting class arcs

As described in Poole & Mackworth, 14.2.3, an arc

\[ \text{c} \rightarrow \text{n} \]

from a class c to n labeled p represents the clause

\[
\text{prop(Ind,p,n)} :\text{- prop(Ind,is-a,c)}. \]

Turning constants p, n and c into variables Att, Val, and Class (respectively), this principle of inheritance generalizes to

(1) \[
\text{prop(Ind,Att,Val)} :\text{- prop(Ind,is-a,Class)},
\]

\[
\text{prop(Class,Att,Val)}. \]

(1) says an individual Ind that is a (member of) Class “inherits” all attribute-values Att, Val of Class.
Non-termination

Assuming $O \Rightarrow C$ means $O$ is-a $C$, the semantic net below says that Fido is a dog, and a dog has 4 legs.

```
  dog  ^legs  4  prop(dog,legs,4).
      ↑
  fido  prop(fido,is-a,dog).
```

Let us form a knowledge base consisting of the two prop facts above and our principle (1) of inheritance. Against this knowledge base, try the queries

```
| ?- prop(fido,legs,4).
| ?- prop(fido,legs,10).
| ?- prop(fido,is-a,cat).
```

Trace the last two queries, observing the similarity between (1) and the rule

```
ancestors(X,Y) :- ancestors(X,Z),
                ancestors(Z,Y).
```
To break looping on ancestor queries that ought to fail, the recursive rule for ancestor is typically formulated as

\[
\text{ancestor}(X,Y) :- \text{parent}(X,Z), \text{ancestor}(Z,Y).
\]

Similarly, to avoid matching the head \text{prop}(\text{Ind}, \text{Att}, \text{Val})\ of (1) with its first assumption \text{prop}(\text{Ind}, \text{is-a}, \text{Class})\, let us rewrite \text{is-a} facts without \text{prop}, reformulating \text{prop}(\text{fido}, \text{is-a}, \text{dog})\ as \text{is-a}(\text{fido}, \text{dog})\, and (1) as

(2) \text{prop}(\text{Ind}, \text{Att}, \text{Val}) :- \text{is-a}(\text{Ind}, \text{Class}),
\quad \text{prop}(\text{Class}, \text{Att}, \text{Val}).

Re-try the queries above, rephrasing the last as \text{is-a}(\text{fido}, \text{cat}).
Inheritance along is-a

Next, add the fact that a dog is a pet to our knowledge base.

```
<table>
<thead>
<tr>
<th>pet</th>
<th>is-a(dog,pet).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>prop(dog,legs,4).</td>
</tr>
<tr>
<td></td>
<td>is-a(fido,dog).</td>
</tr>
</tbody>
</table>
```

Try the query

```
| ?- is-a(fido,pet). |
```

To apply the principle of inheritance to \( \text{Att} = \text{is-a} \), let us break the loop from (1) in another way, splitting prop off between two predicates \text{prim}[itive] and \text{der}[ived]. Use \text{prim} to record explicitly given facts such as

```
| prim(dog,is-a,pet).  prim(dog,legs,4).  prim(fido,is-a,dog). |
```
We reserve \texttt{der} for derived facts, reformulating (2) as
\[\text{der}(OC, Att, Val) :- \text{prim}(OC, Att, Val).\]
(3) \[\text{der}(OC, Att, Val) :- \text{prim}(OC, is-a, Class),\]
\[\quad \text{der}(Class, Att, Val).\]

Try the queries
\[| \text{?- prim(fido, is-a, pet).} \]
\[| \text{?- der(fido, is-a, pet).} \]

Derived, contra primitive
Exceptions over-riding defaults

Now, another dog, Rover, comes along, and bites off one of Fido’s legs.

Try the queries

| ?- der(rover,legs,X). |
| ?- der(fido,legs,3). |
| ?- der(fido,legs,4). |

To block the derivation that Fido has 4 legs, change (3) to

\[\text{(4) } \text{der}(\text{OC}, \text{Att}, \text{Val}) :- \bigoplus \text{prim}(\text{OC}, \text{Att}, _), \text{prim}(\text{OC}, \text{is-a}, \text{Class}), \text{der}(\text{Class}, \text{Att}, \text{Val}).\]
But now re-try the query

\[ \text{der(fido,is-a,pet)}. \]

Unlike legs, \textit{is-a} is a multi-valued attribute. Accordingly, change (4) to

\[ \text{(5) der(OC,Att,Val) :- (\text{prim(OC,Att,\_)} ; \text{multiValued(Att)}),} \]
\[ \text{prim(OC,is-a,Class),} \]
\[ \text{der(Class,Att,Val).} \]

\[ \text{multiValued(is-a).} \]

Re-try the queries above.
Suppose we agreed that pets have 6 legs (e.g. spiders).

\[
\begin{align*}
\text{pet} & \rightarrow 6 \\
\text{rover} & \rightarrow \text{dog} \\
\text{fido} & \rightarrow 3
\end{align*}
\]

How many legs does Rover have? (5) implements the principle “most specific rule wins” where specificity falls with every appeal to an is-a arc. What happens if we were to add an is-a arc from rover to pet? How many legs does Rover get?
Addendum

Revise

\[
\begin{align*}
der&(OC,Att,Val) :- \text{prim}(OC,Att,Val). \\
(4)\quad&der(OC,Att,Val) :- \neg \text{prim}(OC,Att,\_), \\
&\quad\text{prim}(OC,\text{is-a},\text{Class}), \\
&\quad\text{der}(\text{Class},\text{Att},\text{Val}).
\end{align*}
\]

to

\[
\begin{align*}
der&(OC,Att,Val) :- \text{prim}(OC,Att,X), !, X = Val. \\
&\quad\text{der}(OC,Att,Val) :- \text{prim}(OC,\text{is-a},\text{Class}), \\
&\quad\text{der}(\text{Class},\text{Att},\text{Val}).
\end{align*}
\]

Move negation-as-failure from recursive clause of \text{der} to a cut in the base case of \text{der}, avoiding the recomputation of \text{prim}(OC,Att,Val).

And for multi-valued predicates, try changing base clause to

\[
\begin{align*}
der&(OC,Att,Val) :- \text{prim}(OC,Att,Val), \\
&(\text{multiValued}(Att); !).
\end{align*}
\]