the compound wrapper cat[binop[x]]

Johan G. F. Belinfante
2011 December 18

In[1]:= SetDirectory["l:"]; << goedel.11dec17a

:Package Title: goedel.11dec17a 2011 December 17 at 7:30 p.m.
Loading takes about thirteen minutes, half that time due to builtin pauses.
It is now: 2011 Dec 18 at 16:48
Loading Simplification Rules
TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3
weightlimit = 40
Loading completed.
It is now: 2011 Dec 18 at 17:1

summary

The category wrapper cat[x] can be written as \( x \cap \text{category}[x] \). As a corollary it is shown that the compound wrapper cat[binop[x]] can serve as a wrapper for a monoid or empty set.

derivation

The cat[x] wrapper is defined by class-wrapped membership rule.

In[6]:= Begin["Goedel`Private`"];

In[7]:= FirstMatch[class[t_, member[w_, HoldPattern[cat[x_]]]]]
Out[7]= class[t_, member[w_, cat[x_]]] := class[t, and[member[w, x], category[x]]]

Because using the class rules is often slow, especially here with the complex concept of category, it would be nice to have a simpler equivalent characterization. This is accomplished in the first theorem. The derivation is speeded up by clearing the simplify and cond flags.

In[2]:= simplify = False; cond = False;

Theorem. A formula for the cat[x] wrapper.
Theorem. If a category is a binary operation, then it is either empty or a monoid.

Lemma. If \( \text{binop}[x] \) is a category, then the compound wrapper is a binary operation.

Theorem. The compound wrapper is a binary operation.

Theorem. If a category is a binary operation, then it is either empty or a monoid.
The following is a better rewrite rule.

Theorem.  Wrapper introduction rule for the compound wrapper \texttt{cat[binop[x]]}.

In[20]:= \texttt{equiv[member[cat[binop[x]], MONOIDS], not[empty[cat[binop[x]]]]]}

Out[20]= True

In[21]:= \texttt{member[cat[binop[x_]], MONOIDS] := not[equal[0, cat[binop[x]]]]}

Lemma.

In[22]:= \texttt{SubstTest[implies, equal[x, binop[t]],}

\hspace{1em} \texttt{or[equal[x, cat[binop[x]]], not[category[x]], t \rightarrow x] // Reverse}

Out[22]= or[equal[x, cat[binop[x]]], not[category[x]], not[member[x, BINOPS]]] = True

In[23]:= \texttt{(\% / . x \rightarrow x_) / . Equal \rightarrow SetDelayed}

Theorem.  Wrapper removal rule for the compound wrapper \texttt{cat[binop[x]]}.

In[24]:= \texttt{Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],}

\hspace{1em} \texttt{not[implies[p1, p4]], \{p1 \rightarrow member[x, MONOIDS], p2 \rightarrow member[x, BINOPS],}

\hspace{2em} \texttt{p3 \rightarrow category[x], p4 \rightarrow equal[x, cat[binop[x]]]\}] / Reverse}

Out[24]= or[equal[x, cat[binop[x]]], not[member[x, MONOIDS]]] = True

In[25]:= \texttt{(\% / . x \rightarrow x_) / . Equal \rightarrow SetDelayed}

Theorem.  Wrapper removal rule for the compound wrapper \texttt{cat[binop[x]]}.

In[26]:= \texttt{equiv[equal[x, cat[binop[x]]], or[equal[0, x], member[x, MONOIDS]]] // not // not}

Out[26]= True

In[27]:= \texttt{equal[x_, cat[binop[x_]]] := or[equal[0, x], member[x, MONOIDS]]}