composites of constant functions

Johan G. F. Belinfante
2006 June 10

In[1]: = SetDirectory["l: "] << goedel82.09b; << tools.m

:Package Title: goedel82.09b 2006 June 9 at 7:00 p.m.

It is now: 2006 Jun 10 at 6:12

Loading Simplification Rules

TOOLS.M Revised 2006 June 6

weightlimit = 40

summary

The composite of a constant function with any function, in either order, is again a constant function. The composite could be empty. The empty function is considered here to be a constant function:

In[2]: = member[0, CONST]


composite in one order

The formula in one order is an immediate consequence of a rewrite rule for composite[COMPOSE, cross[Id,CART]]:

In[3]: = ImageComp[COMPOSE, cross[Id, CART], cart[FUNS, cart[V, range[SINGLETON]]]] // Reverse

Out[3]= image[COMPOSE, cart[FUNS, CONST]] = CONST

In[4]: = image[COMPOSE, cart[FUNS, CONST]] := CONST

a lemma

One is tempted to try the same strategy for the composite in the other order. The first step is to derive an analog of the rewrite rule used above. This is easy:
The direct analog of the attack used above does not succeed, but a useful lemma is obtained when one replaces \textsc{Funs} with $\text{P}[	ext{Id}]$:

\begin{verbatim}
Out[11] = image[COMPOSE, cart[CONST, P[Id]]] = CONST
In[12] := image[COMPOSE, cart[CONST, P[Id]]] := CONST
\end{verbatim}

**Composite in the other order**

Lemma. The composite of a cartesian product with anything is again a cartesian product.

\begin{verbatim}
In[13] := Map[equal[0, #] &,
   dif[cart[range[CART], x], image[inverse[COMPOSE], range[CART]]]] // VSNormality
Out[13] = subclass[image[COMPOSE, cart[range[CART], x]], range[CART]] = True
In[14] := subclass[image[COMPOSE, cart[range[CART], x_]], range[CART]] := True
\end{verbatim}

Since constant functions are cartesian products, the following corollary is obtained:

\begin{verbatim}
In[15] := SubstTest[implies, andsubclass[u, v], subclass[v, w]], subclass[u, w],
   {u \mapsto cart[CONST, x], v \mapsto cart[range[CART], V],
   w \mapsto image[inverse[COMPOSE], range[CART]]}
Out[15] = subclass[image[COMPOSE, cart[CONST, x]], range[CART]] = True
In[16] := subclass[image[COMPOSE, cart[CONST, x_]], range[CART]] := True
\end{verbatim}

Since the composite of two functions is a function, this is true in particular for constant functions:

\begin{verbatim}
In[17] := SubstTest[implies, andsubclass[u, v], subclass[v, w]], subclass[u, w],
   {u \mapsto cart[CONST, FUNS], v \mapsto cart[FUNS, FUNS], w \mapsto image[inverse[COMPOSE], FUNS]}
In[18] := \% /. Equal \rightarrow SetDelayed
\end{verbatim}

Combining these results, one obtains an inclusion:
The lemma derived in the preceding section yields an inclusion in the opposite direction:

\[
\text{In \[21\]} := \text{SubstTest[\text{implies, subclass[u, v], subclass[image[w, u], image[w, v]],}} \\
\{\text{u \to \text{cart[CONST, P[Id]]}, v \to \text{cart[CONST, FUNS], w \to \text{COMPOSE}}]}}
\]

\[
\text{Out\[21\]} = \text{subclass[\text{CONST, image[COMPOSE, cart[CONST, FUNS]]}] := True}
\]

\[
\text{In\[22\]} := %/. \text{Equal \to SetDelayed}
\]

Combining these two inclusions yields the desired equation:

\[
\text{In\[23\]} := \text{SubstTest[\text{and, subclass[u, v], subclass[v, u],}} \\
\{\text{u \to \text{CONST, v \to image[COMPOSE, cart[CONST, FUNS]]})}}
\]

\[
\text{Out\[23\]} = \text{True := equal[\text{CONST, image[COMPOSE, cart[CONST, FUNS]]]}}
\]

\[
\text{In\[24\]} := \text{image[COMPOSE, cart[CONST, FUNS]] := CONST}
\]