unions of mappings with disjoint domains

Johan G. F. Belinfante
2006 August 19

In[1]:= SetDirectory["1:"]; << goedel84.17b; << tools.m

:Package Title: goedel84.17b 2006 August 17 at 3:20 p.m.

It is now: 2006 Aug 19 at 14:37

Loading Simplification Rules

TOOLS.M Revised 2006 August 15

weightlimit = 40

summary

The union of two functions with disjoint domains is a function. In this notebook it is shown that when \( x \) and \( y \) are disjoint, one can express the class of mappings from \( \text{union}[x,y] \) to \( z \) in terms of the class of mappings from \( x \) to \( z \) and the class of mappings from \( y \) to \( z \). An application of this formula is the theorem that when \( x \) and \( y \) are disjoint, and \( z \) is a set, the set of mappings from \( \text{union}[x,y] \) to \( z \) is equipollent to the cartesian product of the set of mappings from \( x \) to \( z \) and the set of mappings from \( y \) to \( z \).

A novel feature of the derivations in this notebook is that two of the derivations presented contain missing steps that are automatically filled by the rewrite rules of the GOEDEL program. Often when one supplies a complete proof, the sheer number of steps and proposition labels \( p_1, p_2, ... \) in the proof will cause the GOEDEL program to take a long time to verify the validity of the argument. One can of course cut down the number of labels by replacing several hypotheses with a single one in the form of a conjunction \( \text{and} [...] \).

map restrictions

One step of the following proof was deliberately omitted: \text{implies}[\text{and}[p5,p6,p7,p8], p9]. The GOEDEL program was able to supply this missing step. Comments: This derivation takes a little over 34 seconds. An attempt to derive this result without omitting this step was aborted after a minute or two, when the fans came on to cool the CPU.
An inclusion

Lemma.

In[7]:= ImageComp[IMAGE[id[cart[x, V]]], id[P[cart[x, V]]], map[x, y]] // Reverse
Out[7]= image[IMAGE[id[cart[x, V]]], map[x, y]] = map[x, y]

In[8]:= image[IMAGE[id[cart[x_, V]]], map[x_, y_]] := map[x, y]

Lemma.

In[9]:= ImageComp[composite[CUP, cross[IMAGE[id[cart[x, V]]], IMAGE[id[cart[y, V]]]]],
DUP, map[union[x, y, z]]] // Reverse
Out[9]= image[CUP, composite[IMAGE[id[cart[y, V]]],
id[map[union[x, y, z]]], inverse[IMAGE[id[cart[x, V]]]]]] = map[union[x, y, z]]
The number of steps needed in this first lemma is reduced somewhat by only considering map classes of the form \( \text{map}[x, V] \). This derivation also bundles four hypotheses, and deliberately omits one step of the proof: \( \text{implies}[\text{and}[p6, p7, p8], p9] \), relying on rewrite rules to fill this gap. Despite all these measures, the execution time is still 73 seconds.

Removing the variables \( u \) and \( v \) yields:

\[
\text{Map}[[\text{equal}[0, \text{composite}[\text{Id}, \text{complement}[^#]]]], \&, \text{SubstTest}[\text{class}, \text{pair}[u, v]],
\]
The restriction to map classes of the form map[\(x, V\)] is now lifted.

Combining this inclusion with the inclusion in the opposite direction yields the main theorem:

\[
\text{a one-to-one restriction of CUP}
\]
SubstTest[composite, w, id[domain[w]],
  w -> composite[id[P[complement[x]]], inverse[IMAGE[id[x]]],
    complement[composite[SECOND, intersection[composite[inverse[FIRST], IMAGE[id[x]]],
      inverse[CUP]]]]], id[P[x]]] // Reverse

Out[27]= composite[id[P[complement[x]]],
 inverse[IMAGE[id[x]]], complement[composite[SECOND, intersection[
  composite[inverse[FIRST], IMAGE[id[x]]], inverse[CUP]]]]], id[P[x]] = 0

In[28]:= (% /. x_ -> _) /. Equal -> SetDelayed

Lemma.

SubstTest[composite, w, id[domain[w]],
  w -> dif[compositeintersection[composite[inverse[FIRST], IMAGE[id[x]]],
    composite[inverse[SECOND], IMAGE[id[complement[x]]]]]],
  CUP, id[cart[P[x]], P[complement[x]]]]], Id] // Reverse

Out[29]= composite[
    intersection[Di, compositeintersection[composite[inverse[FIRST], IMAGE[id[x]]],
      composite[inverse[SECOND], IMAGE[id[complement[x]]]]]],
    Id[cart[P[x], P[complement[x]]]]], 0

In[30]:= (% /. x_ -> _) /. Equal -> SetDelayed

Lemma.

SubstTest[equal, 0, dif[u, v],
{u -> compositeintersection[composite[inverse[FIRST], IMAGE[id[x]]],
    composite[inverse[SECOND], IMAGE[id[complement[x]]]]],
   CUP, Id[cart[P[x], P[complement[x]]]]], v -> Id] // Reverse

Out[31]= subclass[compositeintersection[composite[inverse[FIRST], IMAGE[id[x]]],
    composite[inverse[SECOND], IMAGE[id[complement[x]]]]]],
    CUP, Id[cart[P[x], P[complement[x]]]]], Id] = True

In[32]:= (% /. x_ -> _) /. Equal -> SetDelayed

Theorem.

SubstTest[implies, subclass[composite[u, v], Id],
  FUNCTION[composite[inverse[v], id[domain[u]]]]],
  {u -> compositeintersection[composite[inverse[FIRST], IMAGE[id[x]]],
    composite[inverse[SECOND], IMAGE[id[complement[x]]]]],
   v -> composite[CUP, Id[cart[P[x], P[complement[x]]]]]]]

Out[33]= FUNCTION[composite[Id[cart[P[x], P[complement[x]]]]], inverse[CUP]]] = True

In[34]:= FUNCTION[composite[Id[cart[P[x], P[complement[x]]]]], inverse[CUP]]] := True

Corollary.
Another corollary of the theorem derived in the preceding section:

Lemma.

Main Theorem.
In[42]: = or[member[pair[cart[map[x_, setpart[z_]], map[y_, setpart[z_]]],
                    map[union[x_, y_], setpart[z_]]], Q], not[equal[0, intersection[x_, y_]]]] := True

The setpart wrapper can be removed:

In[43]: = Map[implies[member[z, w], #] & SubstTest[implies, equal[z, setpart[t]]],
         or[member[pair[cart[map[x, z], map[y, z]],
                    map[union[x, y], z]], Q],
            not[equal[0, intersection[x, y]]], t \[\rightarrow\] z]]

Out[43]= or[member[pair[cart[map[x, z], map[y, z]], map[union[x, y], z]], Q],
        not[equal[0, intersection[x, y]]], not[member[z, w]]] = True

In[44]: = or[member[pair[cart[map[x_, z_], map[y_, z_]], map[union[x_, y_], z_]], Q],
         not[equal[0, intersection[x_, y_]]], not[member[z_, w_]]] := True

Comment. The classes x and y need not be sets, but if either one is a proper class, then the theorem just reduces to the trivial statement that the empty set is equipollent to itself.

In[46]: = implies[not[member[u, V]], equal[0, map[u, v]]]

Out[46]= True