unions of mappings with disjoint domains

Johan G. F. Belinfante
2006 August 19

```
In[1]:= SetDirectory["l:"]; << 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 **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 **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 **p1, p2, ...** 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 **and[...]**.

map restrictions

One step of the following proof was deliberately omitted: **implies[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.

```
In[2]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
          implies[p1, p4], implies[and[p1, p2], p5], implies[and[p1, p3], p6],
          implies[and[p1, p4], p7], implies[p1, p8], not[implies[p1, p9]],
          {p1 → and[member[u, map[x, y]], equal[v, composite[u, id[w]]]], p2 → FUNCTION[u],
           p3 → equal[x, domain[u]], p4 → subclass[range[u], y], p5 → FUNCTION[v], p6 →
            equal[domain[v], intersection[x, w]], p7 → subclass[range[v], y], p8 → member[v, V],
           p9 → member[v, map[intersection[w, x], y]]}]] /. v -> composite[u, id[w]]
```

```
Out[2]= or[member[composite[u, id[w]], map[intersection[w, x], y]],
         not[member[u, map[x, y]]]] == True
```

```
In[3]:= or[member[composite[u_, id[w_]], map[intersection[w_, x_], y_]],
         not[member[u_, map[x_, y_]]]] := True
```

The variable **u** is eliminated as follows:

```
In[4]:= Map[equal[V, #] &,
         dif[map[x, y], image[inverse[IMAGE[id[cart[w, V]]]], map[intersection[w, x], y]]] //
          complement // Normality]
```

```
Out[4]= subclass[image[IMAGE[id[cart[w, V]]], map[x, y]], map[intersection[w, x], y]] == True
```

```
In[5]:= subclass[image[IMAGE[id[cart[w_, V]]], map[x_, y_]],
         map[intersection[w_, x_], y_]] := True
```

Comment: This can not be sharpened to an equation. A simple counterexample is provided by the case **w = 0, x = V**.

```
In[6]:= equal[image[IMAGE[id[cart[w, V]]], map[x, y]],
         map[intersection[w, x], y]] /. {w → 0, x → V}
```

```
Out[6]= False
```

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]
```

```
In[10]:= 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]
```

Lemma.

```
In[11]:= SubstTest[subclass, image[IMAGE[id[cart[x, V]]], map[t, z]],
          map[intersection[x, t], z], t → union[x, y]]

Out[11]= subclass[image[IMAGE[id[cart[x, V]]], map[union[x, y], z]], map[x, z]] == True

In[12]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[13]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> composite[IMAGE[id[cart[y, V]]], id[map[union[x, y], z]],
             inverse[IMAGE[id[cart[x, V]]]]], v -> cart[map[x, z], map[y, z]], w → CUP}]

Out[13]= subclass[map[union[x, y], z], image[CUP, cart[map[x, z], map[y, z]]]] == True

In[14]:= subclass[map[union[x_, y_], z_], image[CUP, cart[map[x_, z_], map[y_, z_]]]] := True
```

---

inclusion in the opposite direction

The number of steps needed in this first lemma is reduced somewhat by only considering map classes of the form **map[x, V]**. This derivation also bundles four hypotheses, and deliberately omits one step of the proof: **implies[and[p6,p7,p8], p9]**, relying on rewrite rules to fill this gap. Despite all these measures, the execution time is still 73 seconds.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p1, p4],
            implies[p1, p5], implies[p1, p6], implies[and[p1, p2, p3, p4, p5], p7],
            implies[and[p1, p3, p5], p8], not[implies[p1, p9]], {p1 → and[equal[w, union[u, v]],
               member[u, map[x, V]], member[v, map[y, V]], disjoint[x, y]], p2 → FUNCTION[u],
             p3 → equal[x, domain[u]], p4 → FUNCTION[v], p5 → equal[y, domain[v]],
             p6 → member[w, V], p7 -> FUNCTION[w], p8 → equal[union[x, y], domain[w]],
             p9 → member[w, map[union[x, y], V]]}]] /. w → union[u, v]

Out[15]= or[member[union[u, v], map[union[x, y], V]], not[equal[0, intersection[x, y]]],
          not[member[u, map[x, V]]], not[member[v, map[y, V]]]] == True

In[16]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Removing the variables **u** and **v** yields:

```
In[17]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class, pair[u, v],
            implies[and[disjoint[x, y], member[u, r], member[v, s]], member[pair[u, v], t]],
            {r → map[x, V], s → map[y, V], t → image[inverse[CUP], map[union[x, y], V]]}]] //
          Reverse

Out[17]= or[not[equal[0, intersection[x, y]]],
          subclass[image[CUP, cart[map[x, V], map[y, V]]], map[union[x, y], V]]] == True
```

```
In[18]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[19]:= ImageComp[CUP, id[cart[P[cart[V, z]], P[cart[V, z]]]], cart[map[x, V], map[y, V]]]

Out[19]= intersection[image[CUP, cart[map[x, V], map[y, V]]], P[cart[V, z]]] ==
          image[CUP, cart[map[x, z], map[y, z]]]

In[20]:= intersection[image[CUP, cart[map[x_, V], map[y_, V]]], P[cart[V, z_]]] :=
          image[CUP, cart[map[x, z], map[y, z]]]
```

The restriction to map classes of the form **map[x, V]** is now lifted.

```
In[21]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> image[CUP, cart[map[x, V], map[y, V]]],
           v -> map[union[x, y], V], w → id[P[cart[V, z]]]}]

Out[21]= or[not[subclass[image[CUP, cart[map[x, V], map[y, V]]], map[union[x, y], V]]],
           subclass[image[CUP, cart[map[x, z], map[y, z]]], map[union[x, y], z]]] == True

In[22]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

In[23]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → disjoint[x, y],
            p2 -> subclass[image[CUP, cart[map[x, V], map[y, V]]], map[union[x, y], V]],
            p3 -> subclass[image[CUP, cart[map[x, z], map[y, z]]], map[union[x, y], z]]}]]

Out[23]= or[not[equal[0, intersection[x, y]]],
           subclass[image[CUP, cart[map[x, z], map[y, z]]], map[union[x, y], z]]] == True

In[24]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

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

```
In[25]:= SubstTest[and, implies[p, subclass[u, v]], subclass[v, u],
           {p → disjoint[x, y],
            u -> image[CUP, cart[map[x, z], map[y, z]]], v -> map[union[x, y], z]}] // Reverse

Out[25]= or[equal[image[CUP, cart[map[x, z], map[y, z]]], map[union[x, y], z]],
           not[equal[0, intersection[x, y]]]] == True

In[26]:= or[equal[image[CUP, cart[map[x_, z_], map[y_, z_]]], map[union[x_, y_], z_]],
           not[equal[0, intersection[x_, y_]]]] := True
```

a one-to-one restriction of CUP

Lemma.

```
In[27]:= 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 → x_) /. Equal → SetDelayed
```

Lemma.

```
In[29]:= SubstTest[composite, w, id[domain[w]],
           w -> dif[composite[intersection[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, composite[intersection[composite[inverse[FIRST], IMAGE[id[x]]],
              composite[inverse[SECOND], IMAGE[id[complement[x]]]]], CUP]],
           id[cart[P[x], P[complement[x]]]]] == 0

In[30]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[31]:= SubstTest[equal, 0, dif[u, v],
           {u -> composite[intersection[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[composite[intersection[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 → x_) /. Equal → SetDelayed
```

Theorem.

```
In[33]:= SubstTest[implies, subclass[composite[u, v], Id],
          FUNCTION[composite[inverse[v], id[domain[u]]]],
          {u -> composite[intersection[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.

```
In[35]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]], FUNCTION[u],
          {u -> composite[id[cart[P[x], P[y]]], inverse[CUP]],
           v -> composite[id[cart[P[x], P[complement[x]]]], inverse[CUP]]}]

Out[35]= or[FUNCTION[composite[id[cart[P[x], P[y]]], inverse[CUP]]],
           not[equal[0, intersection[x, y]]]] == True

In[36]:= or[FUNCTION[composite[id[cart[P[x_], P[y_]]], inverse[CUP]]],
           not[equal[0, intersection[x_, y_]]]] := True
```

equipollence theorem for sets of mappings

Another corollary of the theorem derived in the preceding section:

```
In[37]:= Map[implies[disjoint[x, y], #] &,
          SubstTest[implies, and[subclass[u, v], FUNCTION[v]], FUNCTION[u],
           {u -> composite[id[cart[map[x, z], map[y, z]]], inverse[CUP]],
            v -> composite[id[cart[P[cart[x, V]], P[complement[cart[x, V]]]]], inverse[CUP]]}]]

Out[37]= or[FUNCTION[composite[id[cart[map[x, z], map[y, z]]], inverse[CUP]]],
           not[equal[0, intersection[x, y]]]] == True

In[38]:= or[FUNCTION[composite[id[cart[map[x_, z_], map[y_, z_]]], inverse[CUP]]],
           not[equal[0, intersection[x_, y_]]]] := True
```

Lemma.

```
In[39]:= SubstTest[implies, member[w, BIJ], member[pair[domain[w], range[w]], Q],
          w → composite[CUP, id[cart[map[x, setpart[z]], map[y, setpart[z]]]]]]

Out[39]= or[member[pair[cart[map[x, setpart[z]], map[y, setpart[z]]],
             image[CUP, cart[map[x, setpart[z]], map[y, setpart[z]]]]], Q], not[FUNCTION[
             composite[id[cart[map[x, setpart[z]], map[y, setpart[z]]]], inverse[CUP]]]]] == True

In[40]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Main Theorem.

```
In[41]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p1, p4],
           implies[and[p3, p4], p5], not[implies[p1, p5]], {p1 → disjoint[x, y], p2 -> FUNCTION[
              composite[id[cart[map[x, setpart[z]], map[y, setpart[z]]]], inverse[CUP]]],
            p3 -> member[pair[cart[map[x, setpart[z]], map[y, setpart[z]]],
               image[CUP, cart[map[x, setpart[z]], map[y, setpart[z]]]]], Q],
            p4 -> equal[image[CUP, cart[map[x, setpart[z]], map[y, setpart[z]]]],
              map[union[x, y], setpart[z]]], p5 -> member[pair[
               cart[map[x, setpart[z]], map[y, setpart[z]]], map[union[x, y], setpart[z]]], Q]}]]

Out[41]= 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
```

```
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 → 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
```