disjoint unions and equipollence

Johan G. F. Belinfante
2005 January 6

```
In[1]:= SetDirectory["i:"]; << goedel65.06a; << tools.m

        :Package Title: goedel65.06a             2005 January 6 at 9:40 p.m.

        It is now:  2005 Jan 6 at 23:57

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 December 21

        weightlimit = 40
```

summary

If a pair of disjoint sets are equipollent to another disjoint pair, then their unions are equipollent.

derivation

```
In[2]:= implies[and[ONEONE[x], ONEONE[y], disjoint[domain[x], domain[y]],
           disjoint[range[x], range[y]]], ONEONE[union[x, y]]] // NotNotTest

Out[2]= or[and[FUNCTION[union[x, y]], FUNCTION[union[inverse[x], inverse[y]]]],
          not[equal[0, intersection[domain[x], domain[y]]]],
          not[equal[0, intersection[range[x], range[y]]]],
          not[FUNCTION[x]], not[FUNCTION[y]],
          not[FUNCTION[inverse[x]]], not[FUNCTION[inverse[y]]]] == True

In[3]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[4]:= member[pair[x, y], composite[inverse[IMAGE[SECOND]], z]] // AssertTest

Out[4]= member[pair[x, y], composite[inverse[IMAGE[SECOND]], z]] ==
         and[member[x, V], member[y, V], member[pair[x, range[y]], z]]

In[5]:= member[pair[x_, y_], composite[inverse[IMAGE[SECOND]], z_]] :=
         and[member[x, V], member[y, V], member[pair[x, range[y]], z]]
```

The elimination of the variables depends on the following observation.

```
In[6]:= composite[inverse[DORA], cross[x, y], DORA] // VSNormality // Reverse

Out[6]= intersection[composite[inverse[IMAGE[FIRST]], x, IMAGE[FIRST]],
          composite[inverse[IMAGE[SECOND]], y, IMAGE[SECOND]]] ==
         composite[inverse[DORA], cross[x, y], DORA]

In[7]:= intersection[composite[inverse[IMAGE[FIRST]], x_, IMAGE[FIRST]],
          composite[inverse[IMAGE[SECOND]], y_, IMAGE[SECOND]]] :=
         composite[inverse[DORA], cross[x, y], DORA]
```

The first step in the elimination process is standard:

```
In[8]:= Map[composite[Id, complement[#]] &, SubstTest[class, pair[x, y],
           implies[member[pair[setpart[x], setpart[y]], intersection[u, v]],
            member[pair[setpart[x], setpart[y]], w]],
           {u → intersection[composite[inverse[IMAGE[FIRST]],
                DISJOINT, IMAGE[FIRST]], cart[BIJ, BIJ]], v →
             intersection[composite[inverse[IMAGE[SECOND]], DISJOINT, IMAGE[SECOND]],
              cart[BIJ, BIJ]], w -> image[inverse[CUP], BIJ]}]] // Reverse

Out[8]= composite[id[BIJ], intersection[complement[image[inverse[CUP], BIJ]],
           composite[inverse[DORA], cross[DISJOINT, DISJOINT], DORA]], id[BIJ]] == 0

In[9]:= % /. Equal → SetDelayed
```

There are three occurrences of **BIJ** that need to be eliminated. The first is fairly easy once one rewrites the above as an inclusion.

```
In[10]:= SubstTest[equal, 0, dif[u, v],
           {u → composite[id[BIJ], inverse[DORA], cross[DISJOINT, DISJOINT],
              DORA, id[BIJ]], v → image[inverse[CUP], BIJ]}] // Reverse

Out[10]= subclass[image[CUP, composite[id[BIJ], inverse[DORA],
             cross[DISJOINT, DISJOINT], DORA, id[BIJ]]], BIJ] == True

In[11]:= % /. Equal → SetDelayed
```

Applying the image under **DORA** eliminates one of the **BIJ** classes, replacing it with **Q = image[DORA, BIJ]**.

```
In[12]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> image[CUP, composite[id[BIJ], inverse[DORA],
              cross[DISJOINT, DISJOINT], DORA, id[BIJ]]], v -> BIJ, w → DORA}]

Out[12]= subclass[image[DORA, image[CUP, composite[id[BIJ], inverse[DORA],
             cross[DISJOINT, DISJOINT], DORA, id[BIJ]]]], Q] == True
```

```
In[13]:= % /. Equal → SetDelayed
```

---

pairwise preservation of unions by DORA

The next step is a formula that says that domains and ranges preserve unions. Since the function **DORA** produces an order pair with both the domain and the range, one needs to use **TWIST** to get the unions to be applied componentwise. This is analogous to vector addition; note that in the formula **(a, b) + (c, d) = (a + c, b + d)**, the order of **b** and **c** got swapped.

```
In[14]:= symdif[composite[DORA, CUP],
            composite[cross[CUP, CUP], TWIST, cross[DORA, DORA]]] // VSTriNormality

Out[14]= union[composite[intersection[composite[DORA, CUP],
             composite[complement[cross[CUP, CUP]], TWIST, cross[DORA, DORA]]],
            id[cart[V, V]]], composite[intersection[composite[complement[DORA], CUP],
             composite[cross[CUP, CUP], TWIST, cross[DORA, DORA]]], id[cart[V, V]]]] == 0

In[15]:= % /. Equal → SetDelayed

In[16]:= SubstTest[equal, 0,
          composite[symdif[u, v], id[cart[V, V]]], {u -> composite[DORA, CUP],
           v -> composite[cross[CUP, CUP], TWIST, cross[DORA, DORA]]}]

Out[16]= True == equal[composite[DORA, CUP],
            composite[cross[CUP, CUP], TWIST, cross[DORA, DORA]]]
```

This yields the following formula for union-presevation:

```
In[17]:= composite[DORA, CUP] := composite[cross[CUP, CUP], TWIST, cross[DORA, DORA]]
```

The elimination of the two remaining **BIJ** classes now is easy:

```
In[18]:= Map[subclass[#, Q] &, ImageComp[DORA, CUP, composite[id[BIJ],
             inverse[DORA], cross[DISJOINT, DISJOINT], DORA, id[BIJ]]]]

Out[18]= subclass[composite[CUP, id[DISJOINT],
             cross[Q, Q], id[DISJOINT], inverse[CUP]], Q] == True

In[19]:= subclass[composite[CUP, id[DISJOINT],
             cross[Q, Q], id[DISJOINT], inverse[CUP]], Q] := True
```

a variant of the formula

A general lemma.

```
In[20]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 -> subclass[image[x, y], z],
            p2 → subclass[image[inverse[x], image[x, y]], image[inverse[x], z]],
            p3 → subclass[intersection[domain[x], y], image[inverse[x], z]]}]]

Out[20]= or[not[subclass[image[x, y], z]],
           subclass[intersection[y, domain[x]], image[inverse[x], z]]] == True

In[21]:= or[not[subclass[image[x_, y_], z_]],
           subclass[intersection[y_, domain[x_]], image[inverse[x_], z_]]] := True
```

This general lemma yields in the present case:

```
In[22]:= SubstTest[implies, subclass[image[x, y], z],
          subclass[intersection[y, domain[x]], image[inverse[x], z]],
          {x -> composite[cross[composite[CUP, id[DISJOINT]],
              composite[CUP, id[DISJOINT]]]], y → cross[Q, Q], z → Q}]

Out[22]= subclass[composite[id[DISJOINT], cross[Q, Q], id[DISJOINT]],
           composite[inverse[CUP], Q, CUP]] == True

In[23]:= subclass[composite[id[DISJOINT], cross[Q, Q], id[DISJOINT]],
           composite[inverse[CUP], Q, CUP]] := True
```