CUP and HULL[x]

Johan G. F. Belinfante
2003 September 6

```
In[1]:= << goedel52.s89; << tools.m

        :Package Title: goedel52.s89       2003 September 5 at 4:30 p.m.

        It is now:  2003 Sep 8 at 10:16

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 August 9

        weightlimit = 40
```

summary

This notebook contains a derivation of the fact that if a class **x** is closed under binary unions, then **HULL[x]** preserves binary unions. This generalizes the theorem in topology that the closure of a union of two subsets of a topological space is the union of their closures.

a temporary abbreviation

The following abbreviation saves some writing.

```
In[2]:= hull[x_, y_] := A[intersection[x, image[S, singleton[y]]]]
```

derivation

Lemma

```
In[3]:= Map[or[not[subclass[u, x]], not[subclass[v, y]], #] &,
          SubstTest[implies, subclass[s, t], subclass[image[z, s], image[z, t]],
            {s -> cart[u, v], t -> cart[x, y]}]]

Out[3]= or[not[subclass[u, x]], not[subclass[v, y]],
          subclass[image[z, cart[u, v]], image[z, cart[x, y]]]] == True

In[4]:= or[not[subclass[u_, x_]], not[subclass[v_, y_]],
          subclass[image[z_, cart[u_, v_]], image[z_, cart[x_, y_]]]] := True
```

Corollary of the lemma.

```
In[5]:= SubstTest[implies, and[subclass[s, x], subclass[t, y]],
         subclass[image[w, cart[s, t]], image[w, cart[x, y]]],
           {s -> intersection[u, x], t -> intersection[v, y]}]

Out[5]= subclass[image[w, cart[intersection[u, x], intersection[v, y]]],
          image[w, cart[x, y]]] == True

In[6]:= subclass[image[w_, cart[intersection[u_, x_], intersection[v_, y_]]],
          image[w_, cart[x_, y_]]] := True
```

Lemma.

```
In[7]:= SubstTest[implies, subclass[s, t], subclass[A[t], A[s]],
           {s -> image[CUP, cart[intersection[u, image[S, singleton[y]]],
            intersection[v, image[S, singleton[z]]]]],
          t -> intersection[w, image[S, singleton[union[y, z]]]]}]

Out[7]= or[not[subclass[image[CUP, cart[intersection[u, image[S, singleton[y]]],
              intersection[v, image[S, singleton[z]]]]], w]],
          subclass[A[intersection[w, image[S, singleton[union[y, z]]]]],
           union[A[intersection[u, image[S, singleton[y]]]],
            A[intersection[v, image[S, singleton[z]]]]]] == True

In[8]:= (% /. {u -> u_, v -> v_, w -> w_, x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Lemma.

```
In[9]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
              {p1 -> subclass[image[CUP, cart[u, v]], w],
                  p2 ->
             subclass[image[CUP, cart[intersection[u, image[S, singleton[y]]], intersection[v,
                 image[S, singleton[z]]]]], intersection[w, image[S, singleton[union[y, z]]]]],
                  p3 -> subclass[A[intersection[w, image[S, singleton[union[y, z]]]]],
              union[A[intersection[u, image[S, singleton[y]]]],
               A[intersection[v, image[S, singleton[z]]]]]]}]]

Out[9]= or[not[subclass[image[CUP, cart[u, v]], w]],
          subclass[A[intersection[w, image[S, singleton[union[y, z]]]]],
           union[A[intersection[u, image[S, singleton[y]]]],
            A[intersection[v, image[S, singleton[z]]]]]] == True

In[10]:= (% /. {u -> u_, v -> v_, w -> w_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Lemma:

```
In[11]:= SubstTest[implies, subclass[y, w], subclass[hull[x, y], hull[x, w]], w -> union[y, z]]

Out[11]= subclass[A[intersection[x, image[S, singleton[y]]]],
           A[intersection[x, image[S, singleton[union[y, z]]]]]] == True

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

Main result.

```
In[13]:= SubstTest[and, implies[p, subclass[u, v]], subclass[v, u],
            {p -> subclass[image[CUP, cart[x, x]], x],
           u -> hull[x, union[y, z]], v -> union[hull[x, y], hull[x, z]]}] // Reverse

Out[13]= or[equal[A[intersection[x, image[S, singleton[union[y, z]]]]],
           union[A[intersection[x, image[S, singleton[y]]]],
            A[intersection[x, image[S, singleton[z]]]]]],
          not[subclass[image[CUP, cart[x, x]], x]]] == True
```

```
In[14]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Restatement of the main result

```
In[15]:= implies[subclass[image[CUP, cart[x, x]], x],
           equal[hull[x, union[y, z]], union[hull[x, y], hull[x, z]]]]

Out[15]= True
```

eliminating the variables y and z

The hypothesis in the main result will be imposed by composing with the identity restricted to the following class:

```
In[16]:= class[w, subclass[image[CUP, cart[x, x]], x]]

Out[16]= complement[image[V, intersection[complement[x], image[CUP, cart[x, x]]]]]
```

The elimination of **y** and **z** is implicitly accomplished as follows:

```
In[17]:= composite[
           id[complement[image[V, intersection[complement[x], image[CUP, cart[x, x]]]]]],
           symdif[composite[HULL[x], CUP],
            composite[CUP, cross[HULL[x], HULL[x]]]]] // VSTriNormality

Out[17]= union[composite[
            id[complement[image[V, intersection[complement[x], image[CUP, cart[x, x]]]]]],
            intersection[composite[CUP, cross[HULL[x], HULL[x]]],
             composite[complement[HULL[x]], CUP]], id[cart[V, V]]], composite[
            id[complement[image[V, intersection[complement[x], image[CUP, cart[x, x]]]]]],
            intersection[composite[complement[CUP], cross[HULL[x], HULL[x]]],
             composite[HULL[x], CUP]], id[cart[V, V]]]] == 0

In[18]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The final result is:

```
In[19]:= SubstTest[equal, 0, composite[id[w], symdif[u, v], id[z]],
            {u -> composite[HULL[x], CUP],
                v -> composite[CUP, cross[HULL[x], HULL[x]]],
                w ->
             complement[image[V, intersection[complement[x], image[CUP, cart[x, x]]]]],
            z -> cart[V, V]}] // Reverse

Out[19]= or[equal[composite[CUP, cross[HULL[x], HULL[x]]], composite[HULL[x], CUP]],
           not[subclass[image[CUP, cart[x, x]], x]]] == True

In[20]:= or[equal[composite[CUP, cross[HULL[x_], HULL[x_]]], composite[HULL[x_], CUP]],
           not[subclass[image[CUP, cart[x_, x_]], x_]]] := True
```

the case of ADJOIN

There is an existing rule that will be removed:

```
In[21]:= composite[ADJOIN[x], CUP]

Out[21]= composite[CUP, cross[Id, ADJOIN[x]]]

In[22]:= composite[ADJOIN[x_], CUP] =.
```

The theorem proved implies:

```
In[23]:= SubstTest[implies, subclass[image[CUP, cart[x, x]], x],
           equal[composite[CUP, cross[HULL[x], HULL[x]]], composite[HULL[x], CUP]],
             x -> image[S, singleton[z]]]

Out[23]= equal[composite[CUP, cross[ADJOIN[z], ADJOIN[z]]], composite[ADJOIN[z], CUP]] == True
```

One can actually prove a more general result:

```
In[24]:= symdif[composite[ADJOIN[union[x, y]], CUP],
           composite[CUP, cross[ADJOIN[x], ADJOIN[y]]]] // VSNormality

Out[24]= union[intersection[composite[CUP, cross[ADJOIN[x], ADJOIN[y]]],
            composite[Di, ADJOIN[union[x, y]], CUP]],
           intersection[composite[ADJOIN[union[x, y]], CUP],
            composite[complement[CUP], cross[ADJOIN[x], ADJOIN[y]]]]] == 0

In[25]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

It is not clear how one should orient this rule:

```
In[26]:= SubstTest[equal, 0, symdif[u, v], {u -> composite[ADJOIN[union[x, y]], CUP],
             v -> composite[CUP, cross[ADJOIN[x], ADJOIN[y]]]}] // Reverse

Out[26]= equal[composite[CUP, cross[ADJOIN[x], ADJOIN[y]]],
           composite[ADJOIN[union[x, y]], CUP]] == True
```

some other examples

```
In[27]:= SubstTest[implies, subclass[image[CUP, cart[z, z]], z],
           equal[composite[CUP, cross[HULL[z], HULL[z]]], composite[HULL[z], CUP]],
             z -> P[x]]

Out[27]= equal[composite[CUP, id[cart[P[x], P[x]]]], composite[id[P[x]], CUP]] == True

In[28]:= SubstTest[implies, subclass[image[CUP, cart[z, z]], z],
           equal[composite[CUP, cross[HULL[z], HULL[z]]], composite[HULL[z], CUP]],
             z -> SYM]

Out[28]= equal[composite[CUP, cross[HULL[SYM], HULL[SYM]]], composite[HULL[SYM], CUP]] == True

In[29]:= SubstTest[implies, subclass[image[CUP, cart[z, z]], z],
           equal[composite[CUP, cross[HULL[z], HULL[z]]], composite[HULL[z], CUP]],
             z -> REGULAR]

Out[29]= equal[composite[CUP, id[cart[REGULAR, REGULAR]]], composite[id[REGULAR], CUP]] == True

In[30]:= SubstTest[implies, subclass[image[CUP, cart[z, z]], z],
           equal[composite[CUP, cross[HULL[z], HULL[z]]], composite[HULL[z], CUP]],
             z -> FINITE]

Out[30]= equal[composite[CUP, id[cart[FINITE, FINITE]]], composite[id[FINITE], CUP]] == True
```

a corollary and a comment

```
In[31]:= SubstTest[implies, equal[u, v], equal[range[u], range[v]],
             {u -> composite[HULL[x], CUP], v -> composite[CUP, cross[HULL[x], HULL[x]]]}]

Out[31]= or[equal[fix[HULL[x]], image[CUP, cart[fix[HULL[x]], fix[HULL[x]]]]],
           not[equal[composite[CUP, cross[HULL[x], HULL[x]]], composite[HULL[x], CUP]]]] == True

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

In[33]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], implies[p3, p4], not[implies[p1, p4]],
                 {p1 -> subclass[image[CUP, cart[x, x]], x],
            p2 -> equal[composite[CUP, cross[HULL[x], HULL[x]]], composite[HULL[x], CUP]],
                       p3 -> equal[fix[HULL[x]], image[CUP, cart[fix[HULL[x]], fix[HULL[x]]]]],
                       p4 ->
             subclass[image[CUP, cart[fix[HULL[x]], fix[HULL[x]]]], fix[HULL[x]]]}]]

Out[33]= or[not[subclass[image[CUP, cart[x, x]], x]],
           subclass[image[CUP, cart[fix[HULL[x]], fix[HULL[x]]]], fix[HULL[x]]]] == True

In[34]:= or[not[subclass[image[CUP, cart[x_, x_]], x_]],
           subclass[image[CUP, cart[fix[HULL[x_]], fix[HULL[x_]]]], fix[HULL[x_]]]] := True
```

When **x** is a set, this result reduces to the following statement that had been derived earlier:

```
In[35]:= implies[subclass[image[CUP, cart[x, x]], x],
          subclass[image[CUP, cart[Aclosure[x], Aclosure[x]]], Aclosure[x]]]

Out[35]= True
```

It is currently not yet known whether **Aclosure[x]** and **fix[HULL[x]]** are also identical when **x** is a proper class.