interiors of intersections

Johan G. F. Belinfante
2003 August 30

```
In[1]:= << goedel52.s84; << tools.m

        :Package Title: goedel52.s84       2003 August 30 at 3:30 p.m.

        It is now:  2003 Sep 8 at 9:50

        Loading Simplification Rules

        TOOLS.M                      Revised 2003 August 9

        weightlimit = 40
```

summary

This notebook contains a generalization of a theorem in topology that interiors of intersections are intersections of interiors. The only hypothesis on the class **x** is that be closed under binary intersections. A particular case derived previously served as a motivation:

```
In[2]:= composite[HC, CAP]

Out[2]= composite[CAP, cross[HC, HC]]
```

temporary notations

The following temporary abbreviation is convenient:

```
In[3]:= core[x_, y_] := U[intersection[x, P[y]]]
```

A basic property of **core[x,y]** is:

```
In[4]:= SubstTest[implies, member[z, w], subclass[z, U[w]], w -> intersection[x, P[y]]]

Out[4]= or[not[member[z, x]], not[subclass[z, y]], subclass[z, U[intersection[x, P[y]]]]] == True

In[5]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Restatement:

```
In[6]:= implies[and[member[z, x], subclass[z, y]], subclass[z, core[x, y]]]

Out[6]= True
```

Note that cores are unchanged when one replaces **x** by its **Uclosure**:

```
In[7]:= core[Uclosure[x], y] == core[x, y]

Out[7]= True
```

On account of this, one obtains a variant of the preceding theorem:

```
In[8]:= SubstTest[implies, and[member[z, w], subclass[z, y]],
          subclass[z, core[w, y]], w -> Uclosure[x]]

Out[8]= or[not[member[z, Uclosure[x]]],
           not[subclass[z, y]], subclass[z, U[intersection[x, P[y]]]]] == True

In[9]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Cores are monotone:

```
In[10]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
             {u -> P[y], v -> P[z], w -> composite[inverse[E], id[x]]}]

Out[10]= or[not[subclass[y, z]],
            subclass[U[intersection[x, P[y]]], U[intersection[x, P[z]]]]] == True

In[11]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Restatement:

```
In[12]:= implies[subclass[y, z], subclass[core[x, y], core[x, z]]]

Out[12]= True
```

In particular, one obtains the following corollary about cores of intersections:

```
In[13]:= SubstTest[implies, subclass[w, z],
           subclass[core[x, w], core[x, z]], w -> intersection[y, z]]

Out[13]= subclass[U[intersection[x, P[intersection[y, z]]]], U[intersection[x, P[z]]]] == True

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

This holds without any hypothesis on **x**.

```
In[15]:= subclass[core[x, intersection[y, z]], intersection[core[x, y], core[x, z]]]

Out[15]= True
```

theorems

If **x** is closed under binary intersection, then so is **Uclosure[x]**.

```
In[16]:= implies[subclass[image[CAP, cart[x, x]], x],
           subclass[image[CAP, cart[Uclosure[x], Uclosure[x]]], Uclosure[x]]]

Out[16]= True
```

The core of a set is a set.

```
In[17]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
             {u -> core[x, y], v -> y}]

Out[17]= or[member[intersection[x, P[y]], V], not[member[y, V]]] == True

In[18]:= or[member[intersection[x_, P[y_]], V], not[member[y_, V]]] := True
```

The following observation is useful to introduce variables:

```
In[19]:= subclass[cart[x, x], image[inverse[CAP], x]]

Out[19]= subclass[image[CAP, cart[x, x]], x]
```

Closure under binary intersections is reformulated with variables:

```
In[20]:= SubstTest[implies, and[member[r, s], subclass[s, t]], member[r, t],
             {r -> pair[u, v], s -> cart[x, x], t -> image[inverse[CAP], x]}] // MapNotNot

Out[20]= or[member[intersection[u, v], x], not[member[u, x]],
           not[member[v, x]], not[subclass[image[CAP, cart[x, x]], x]]] == True

In[21]:= (% /. {u -> u_, v -> v_, x -> x_}) /. Equal -> SetDelayed
```

Lemma:

```
In[22]:= SubstTest[implies, member[u, P[v]], member[U[u], Uclosure[v]],
             {u -> intersection[x, P[y]], v -> x}]

Out[22]= or[member[U[intersection[x, P[y]]], Uclosure[x]],
           not[member[intersection[x, P[y]], V]]] == True

In[23]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

The core of a set belongs to the **Uclosure** of the class **x**.

```
In[24]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
                   {p1 -> member[y, V], p2 -> member[intersection[x, P[y]], V],
              p3 -> member[U[intersection[x, P[y]]], Uclosure[x]]}]]

Out[24]= or[member[U[intersection[x, P[y]]], Uclosure[x]], not[member[y, V]]] == True

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

Restatement:

```
In[26]:= implies[member[y, V], member[core[x, y], Uclosure[x]]]

Out[26]= True
```

Lemma.

```
In[27]:= SubstTest[implies, and[subclass[u, y], subclass[v, z]],
           subclass[intersection[u, v], intersection[y, z]],
             {u -> core[x, y], v -> core[x, z]}]

Out[27]= and[subclass[intersection[U[intersection[x, P[y]]], U[intersection[x, P[z]]]], y],
             subclass[intersection[U[intersection[x, P[y]]], U[intersection[x, P[z]]]], z]] == True

In[28]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Lemma.

```
In[29]:= SubstTest[implies,
          and[member[u, Uclosure[x]], subclass[u, v]], subclass[u, core[x, v]],
            {u -> intersection[core[x, y], core[x, z]], v -> intersection[y, z]}]

Out[29]= or[not[member[
             intersection[U[intersection[x, P[y]]], U[intersection[x, P[z]]]], Uclosure[x]]],
           subclass[intersection[U[intersection[x, P[y]]], U[intersection[x, P[z]]]],
            U[intersection[x, P[intersection[y, z]]]]] == True

In[30]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Theorem. If **x** is closed under intersections, then cores preserve intersections.

```
In[31]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
           implies[p5, p6], implies[and[p3, p4, p6], p7], implies[p7, p8],
                 not[implies[and[p1, p2, p5], p8]],
             {p1 -> member[y, V], p2 -> member[z, V],
            p3 -> member[core[x, y], Uclosure[x]], p4 -> member[core[x, z], Uclosure[x]],
                 p5 -> subclass[image[CAP, cart[x, x]], x],
            p6 -> subclass[image[CAP, cart[Uclosure[x], Uclosure[x]]], Uclosure[x]],
                 p7 -> member[intersection[core[x, y], core[x, z]], Uclosure[x]],
              p8 ->
             subclass[intersection[core[x, y], core[x, z]], core[x, intersection[y, z]]]}]]

Out[31]= or[not[member[y, V]], not[member[z, V]], not[subclass[image[CAP, cart[x, x]], x]],
           subclass[intersection[U[intersection[x, P[y]]], U[intersection[x, P[z]]]],
            U[intersection[x, P[intersection[y, z]]]]] == True

In[32]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

The inclusion can be strengthened to an equality.

```
In[33]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
            {p -> and[member[y, V], member[z, V], subclass[image[CAP, cart[x, x]], x]],
                u -> intersection[core[x, y], core[x, z]], v -> core[x, intersection[y, z]]}] //
          Reverse

Out[33]= or[equal[intersection[U[intersection[x, P[y]]], U[intersection[x, P[z]]]],
            U[intersection[x, P[intersection[y, z]]]]], not[member[y, V]],
           not[member[z, V]], not[subclass[image[CAP, cart[x, x]], x]]] == True

In[34]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

The variables **y** and **z** can be eliminated:

```
In[35]:= Map[composite[Id, complement[#]] &, SubstTest[class, pair[y, z],
            or[equal[intersection[U[intersection[x, P[y]]], U[intersection[x, P[z]]]],
              U[intersection[x, P[intersection[y, z]]]]], not[member[y, V]],
             not[member[z, V]], not[p]], p -> subclass[image[CAP, cart[x, x]], x]]] // Reverse

Out[35]= composite[
           id[complement[image[V, intersection[complement[x], image[CAP, cart[x, x]]]]]],
           complement[fix[
             composite[inverse[CAP], inverse[CORE[x]], S, CAP, cross[CORE[x], CORE[x]]]]]] == 0

In[36]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The **fix** can be eliminated:

```
In[37]:= SubstTest[equal, 0, fix[composite[inverse[u], complement[v]]],
           {u -> composite[CORE[x], CAP], v -> composite[S, CAP, cross[CORE[x], CORE[x]]]}]

Out[37]= subclass[cart[V, V],
          fix[composite[inverse[CAP], inverse[CORE[x]], S, CAP, cross[CORE[x], CORE[x]]]]] ==
         subclass[composite[CORE[x], CAP], composite[S, CAP, cross[CORE[x], CORE[x]]]]

In[38]:= subclass[cart[V, V], fix[
           composite[inverse[CAP], inverse[CORE[x_]], S, CAP, cross[CORE[x_], CORE[x_]]]]] :=
         subclass[composite[CORE[x], CAP], composite[S, CAP, cross[CORE[x], CORE[x]]]]

In[39]:= SubstTest[equal, 0, composite[id[complement[u]],
           fix[composite[inverse[funpart[v]], complement[S], funpart[w]]]],
            {u -> image[V, intersection[complement[x], image[CAP, cart[x, x]]]],
                v -> composite[CORE[x], CAP], w -> composite[CAP, cross[CORE[x], CORE[x]]]}] //
         Reverse

Out[39]= or[not[subclass[image[CAP, cart[x, x]], x]],
          subclass[composite[CORE[x], CAP], composite[S, CAP, cross[CORE[x], CORE[x]]]]] == True

In[40]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Some additional work is needed to eliminate the subclass relation **S** in this result.

```
In[41]:= dif[composite[CORE[x], CAP],
           composite[inverse[S], CAP, cross[CORE[x], CORE[x]]]] // VSNormality

Out[41]= union[intersection[composite[CORE[x], CAP],
           composite[complement[inverse[S]], CORE[x], FIRST]], intersection[
           composite[CORE[x], CAP], composite[complement[inverse[S]], CORE[x], SECOND]]] == 0

In[42]:= (% /. x -> x_) /. Equal -> SetDelayed

In[43]:= SubstTest[equal, 0, dif[u, v], {u -> composite[CORE[x], CAP],
           v -> composite[inverse[S], CAP, cross[CORE[x], CORE[x]]]}] // Reverse

Out[43]= subclass[composite[CORE[x], CAP],
          composite[inverse[S], CAP, cross[CORE[x], CORE[x]]]] == True

In[44]:= (% /. x -> x_) /. Equal -> SetDelayed
```

An inclusion of functions:

```
In[45]:= Map[implies[subclass[image[CAP, cart[x, x]], x], #] &,
          SubstTest[subclass, u, intersection[v, w],
            {u -> composite[CORE[x], CAP], v -> composite[S, CAP, cross[CORE[x], CORE[x]]],
           w -> composite[inverse[S], CAP, cross[CORE[x], CORE[x]]]}]]

Out[45]= or[not[subclass[image[CAP, cart[x, x]], x]],
          subclass[composite[CORE[x], CAP], composite[CAP, cross[CORE[x], CORE[x]]]]] == True

In[46]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Strengthening to an equality:

```
In[47]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]],
            {u -> composite[CORE[x], CAP], v -> composite[CAP, cross[CORE[x], CORE[x]]]}]

Out[47]= or[equal[composite[CAP, cross[CORE[x], CORE[x]]], composite[CORE[x], CAP]], not[
           subclass[composite[CORE[x], CAP], composite[CAP, cross[CORE[x], CORE[x]]]]]] == True

In[48]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The main result:

```
In[49]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
              {p1 -> subclass[image[CAP, cart[x, x]], x],
            p2 -> subclass[composite[CORE[x], CAP], composite[CAP, cross[CORE[x], CORE[x]]]],
             p3 -> equal[composite[CORE[x], CAP], composite[CAP, cross[CORE[x], CORE[x]]]]}]]

Out[49]= or[equal[composite[CAP, cross[CORE[x], CORE[x]]], composite[CORE[x], CAP]],
           not[subclass[image[CAP, cart[x, x]], x]]] == True

In[50]:= or[equal[composite[CAP, cross[CORE[x_], CORE[x_]]], composite[CORE[x_], CAP]],
           not[subclass[image[CAP, cart[x_, x_]], x_]]] := True
```

comment

In the case **x = FULL** one has

```
In[51]:= CORE[FULL]

Out[51]= HC

In[52]:= image[CAP, cart[FULL, FULL]]

Out[52]= FULL

In[53]:= Uclosure[FULL]

Out[53]= FULL

In[54]:= composite[HC, CAP]

Out[54]= composite[CAP, cross[HC, HC]]
```

The class **FULL** fails to be a topology only because it is a proper class. The class **OMEGA** of all ordinals provides a similar example.

```
In[55]:= composite[CORE[OMEGA], CAP] == composite[CAP, cross[CORE[OMEGA], CORE[OMEGA]]]

Out[55]= True
```

corollaries

If **x** is **Aclosed**, then it is **CAPclosed**. For this case one can obtain an unconditional rule:

```
In[56]:= SubstTest[implies, subclass[image[CAP, cart[y, y]], y], equal[
           composite[CORE[y], CAP], composite[CAP, cross[CORE[y], CORE[y]]]], y -> Aclosure[x]]

Out[56]= equal[composite[CAP, cross[CORE[Aclosure[x]], CORE[Aclosure[x]]]],
           composite[CORE[Aclosure[x]], CAP]] == True

In[57]:= composite[CORE[Aclosure[x_]], CAP] :=
          composite[CAP, cross[CORE[Aclosure[x]], CORE[Aclosure[x]]]]
```

This formula enables one to establish many special cases; for example:

```
In[58]:= SubstTest[composite, CORE[Aclosure[y]], CAP, y -> invar[x]]

Out[58]= composite[CORE[invar[x]], CAP] == composite[CAP, cross[CORE[invar[x]], CORE[invar[x]]]]

In[59]:= composite[CORE[invar[x_]], CAP] :=
          composite[CAP, cross[CORE[invar[x]], CORE[invar[x]]]]

In[60]:= SubstTest[composite, CORE[Aclosure[y]], CAP, y -> binclosed[x]]

Out[60]= composite[CORE[binclosed[x]], CAP] ==
          composite[CAP, cross[CORE[binclosed[x]], CORE[binclosed[x]]]]

In[61]:= composite[CORE[binclosed[x_]], CAP] :=
          composite[CAP, cross[CORE[binclosed[x]], CORE[binclosed[x]]]]

In[62]:= SubstTest[composite, CORE[Aclosure[y]], CAP, y -> SYM]

Out[62]= composite[CORE[SYM], CAP] == composite[CAP, cross[CORE[SYM], CORE[SYM]]]

In[63]:= composite[CORE[SYM], CAP] := composite[CAP, cross[CORE[SYM], CORE[SYM]]]
```

The symmetric core of **x** is:

```
In[64]:= core[SYM, x]

Out[64]= intersection[x, inverse[x]]
```

The following formulas express this fact:

```
In[65]:= lambda[x, intersection[x, inverse[x]]]

Out[65]= CORE[SYM]

In[66]:= composite[CAP, id[IMAGE[SWAP]], inverse[FIRST]]

Out[66]= CORE[SYM]
```