relative topologies for subspaces

Johan G. F. Belinfante
2014 April 9

```
In[1]:= SetDirectory["l:"]; << goedel.14apr07a
```

```
:Package Title: goedel.14apr07a                    2014 April 7 at 5:35 p.m.

Loading takes about seventeen minutes, half that time due to builtin pauses.

It is now:  2014 Apr 9 at 6:57

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2014 Apr 9 at 7:14
```

summary

The relative topology for a subspace **x ⊂ U[t]** of a space with topology **t** is **image[IMAGE[id[x]], t]**. It is shown that the function that takes a subspace to its relative topology is one-to-one. A formula is derived for the result of replacing **t** with the relative topology for a subspace. Rewrite rules are derived for the set of relative topologies for a collection of subspaces. All of the results derived in this notebook are actually more general because no topology axioms are ever used. The class **t** can be any collection of sets.

relative topologies for subspaces

The function that takes **x** to **image[IMAGE[id[x]], t]** is:

```
In[2]:= VERTSECT[reify[x, image[IMAGE[id[x]], t]]]
Out[2]= VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]
```

Theorem.

```
In[3]:= composite[BIGCUP, VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]] //
         RelnNormality
Out[3]= composite[BIGCUP, VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]] ==
         IMAGE[id[U[t]]]
```

```
In[4]:= composite[BIGCUP, VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]]] :=
         IMAGE[id[U[t]]]
```

Theorem.

```
In[5]:= SubstTest[implies, equal[t, domain[x]], implies[subclass[composite[x, y], Id],
           FUNCTION[composite[inverse[y], id[t]]], t → V] // Reverse

Out[5]= or[FUNCTION[inverse[y]],
          not[equal[V, domain[x]]], not[subclass[composite[x, y], Id]]] == True

In[6]:= or[FUNCTION[inverse[y_]], not[equal[V, domain[x_]]],
          not[subclass[composite[x_, y_], Id]]] := True
```

Theorem. The correspondence between subspaces and subspace topologies is one-to-one.

```
In[7]:= SubstTest[implies, and[equal[domain[x], V], subclass[composite[x, y], Id]],
          FUNCTION[inverse[y]], {x → BIGCUP, y -> composite[
             VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], id[P[U[t]]]]}] // Reverse

Out[7]= FUNCTION[composite[id[P[U[t]]],
           inverse[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]]] == True

In[8]:= FUNCTION[composite[id[P[U[t_]]],
           inverse[VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]]]] := True
```

subspaces of subspaces

In this section a formula is derived for the result of replacing **t** with a subspace topology in the function **VERTSECT[CAP ∘ id[V × t] ∘ inverse[FIRST]]**.

Lemma.

```
In[9]:= Map[empty[composite[Id, complement[#]]] &,
         SubstTest[class, pair[u, v], implies[member[pair[u, v], y], member[pair[u, v], z]],
          {y -> composite[IMAGE[IMAGE[id[x]]],
             VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]],
           z -> VERTSECT[composite[CAP, id[cart[V, image[IMAGE[id[x]], t]]], inverse[FIRST]]]}]]

Out[9]= subclass[composite[IMAGE[IMAGE[id[x]]],
           VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]],
          VERTSECT[composite[CAP, id[cart[V, image[IMAGE[id[x]], t]]], inverse[FIRST]]]] == True

In[10]:= (% /. {x → x_, t → t_}) /. Equal → SetDelayed
```

Theorem. Replacing **t** with a subspace topology in the function **VERTSECT[CAP ∘ id[V × t] ∘ inverse[FIRST]]**.

```
In[11]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]], {u -> composite[IMAGE[IMAGE[id[x]]],
              VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]], v -> VERTSECT[
              composite[CAP, id[cart[V, image[IMAGE[id[x]], t]]], inverse[FIRST]]]}] // Reverse

Out[11]= equal[composite[IMAGE[IMAGE[id[x]]],
            VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]],
           VERTSECT[composite[CAP, id[cart[V, image[IMAGE[id[x]], t]]], inverse[FIRST]]]] == True

In[12]:= VERTSECT[composite[CAP, id[cart[V, image[IMAGE[id[x_]], t_]]], inverse[FIRST]]] :=
          composite[IMAGE[IMAGE[id[x]]],
           VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]]
```

Corollary. A simplification rule.

```
In[13]:= SubstTest[VERTSECT,
          composite[CAP, id[cart[V, image[IMAGE[id[x]], t]]], inverse[FIRST]], x → U[t]]

Out[13]= composite[IMAGE[IMAGE[id[U[t]]]],
           VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]] ==
          VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]

In[14]:= composite[IMAGE[IMAGE[id[U[t_]]]],
           VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]]] :=
          VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]
```

Theorem. Another simplification rule.

```
In[15]:= ImageComp[IMAGE[IMAGE[id[x]]], IMAGE[IMAGE[id[U[t]]]],
          VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]]

Out[15]= image[IMAGE[IMAGE[id[intersection[x, U[t]]]]],
           VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]] ==
          image[IMAGE[IMAGE[id[x]]], image[IMAGE[IMAGE[id[U[t]]]],
            VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]]]

In[16]:= image[IMAGE[IMAGE[id[intersection[x_, U[t_]]]]],
           VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]]] :=
          image[IMAGE[IMAGE[id[x]]], image[IMAGE[IMAGE[id[U[t]]]],
            VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]]]
```

Theorem. Yet another simplification rule.

```
In[17]:= Assoc[id[P[P[U[t]]]], IMAGE[IMAGE[id[U[t]]]],
          VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]]

Out[17]= composite[id[P[P[U[t]]]], VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]] ==
          VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]

In[18]:= composite[id[P[P[U[t_]]]],
           VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]]] :=
          VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]
```

relative topologies for sets of subspaces

If **x** is a collection of subspaces of a space with topology **t**, then the class of relative topologies for the collection **x** of subspaces is

image[VERTSECT[CAP ∘ id[V × t] ∘ inverse[FIRST]], x].

In this section, rewrite rules for this class are derived.

Lemma. If the subspace topology for a subspace **x** of a topological space has a certain property, and if **x ⊂ y**, where y is a subspace, then the same property holds for the subspace topology **x** considered as a subspace of **y**.

```
In[19]:= SubstTest[implies, and[equal[x, w], member[image[IMAGE[id[x]], t], z]],
           member[image[IMAGE[id[w]], t], z], w → intersection[x, y]] // Reverse

Out[19]= or[member[image[IMAGE[id[intersection[x, y]]], t], z],
           not[member[image[IMAGE[id[x]], t], z]], not[subclass[x, y]]] == True

In[20]:= or[member[image[IMAGE[id[intersection[x_, y_]]], t_], z_],
           not[member[image[IMAGE[id[x_]], t_], z_]], not[subclass[x_, y_]]] := True
```

Lemma.

```
In[21]:= Map[not, SubstTest[and, implies[and[p0, p1], p5],
            implies[and[p0, p2], p3], implies[and[p3, p4, p5], p6],
            not[implies[and[p0, p1, p2, p4], p6]], {p0 → member[w, x], p1 → subclass[x,
              image[inverse[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]], z]],
             p2 → subclass[x, P[y]], p3 → subclass[w, y],
             p4 → equal[s, image[IMAGE[id[y]], t]], p5 -> member[w,
               image[inverse[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]]], z]],
             p6 -> member[w, image[inverse[VERTSECT[composite[CAP,
                   id[cart[V, s]], inverse[FIRST]]]], z]]}]] // Reverse

Out[21]= or[member[image[IMAGE[id[w]], s], z],
           not[equal[s, image[IMAGE[id[y]], t]]], not[member[w, x]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], z]],
           not[subclass[U[x], y]]] == True

In[22]:= (% /. {s → s_, t → t_, w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[23]:= Map[equal[V, domain[#]] &,
           SubstTest[reify, w, case[or[member[image[IMAGE[id[w]], s], z],
              not[equal[s, image[IMAGE[id[y]], t]]], not[member[w, x]], not[
               subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], z]],
              not[subclass[u, y]]]], u → U[x]]] /. {s -> image[IMAGE[id[y]], t],
           z -> image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]}

Out[23]= or[not[subclass[U[x], y]], subclass[image[IMAGE[IMAGE[id[y]]],
             image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]],
            image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]]] == True

In[24]:= or[not[subclass[U[x_], y_]], subclass[image[IMAGE[IMAGE[id[y_]]],
             image[VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]], x_]],
            image[VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]], x_]]] := True
```

Corollary. The special case $\mathbf{y = U[x]}$.

```
In[25]:= SubstTest[implies, subclass[U[x], y], invariant[IMAGE[IMAGE[id[y]]], image[
             VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]], y → U[x]] // Reverse

Out[25]= subclass[image[IMAGE[IMAGE[id[U[x]]]],
             image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]],
            image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]] == True

In[26]:= (% /. {x → x_, t → t_}) /. Equal → SetDelayed

In[27]:= equal[image[IMAGE[IMAGE[id[U[x]]]],
            image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]],
           image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]]

Out[27]= True

In[28]:= image[IMAGE[IMAGE[id[U[x_]]]],
            image[VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]], x_]] :=
           image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x]
```