unions of nondisjoint connected subspaces

Johan G. F. Belinfante
2014 April 13

```
In[1]:= SetDirectory["l:"]; << goedel.14apr09a
```

```
:Package Title: goedel.14apr09a                    2014 April 9 at 6:25 p.m.

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

It is now:  2014 Apr 13 at 13:17

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 13 at 13:34
```

summary

If a collection of connected subspaces has a point in common, then its union is connected. This result is derived not just for topologies, but for any collection of sets **t** that holds the empty set and its sum class. Since it is convenient to use the traditional terminology used in topology, the notes accompanying the derivations generally pretend that one is dealing with a topology whether or not **t** actually does satisfy the axioms for a topology.

derivation

The class of all connected subspaces of a space with topology **t** is **image[inverse[VERTSECT[CAP ∘ id[V × t] ∘ inverse[-FIRST]]], CONNECTED]] ⋂ P[U[t]]**. The results to be derived are concerned with unions of connected subspaces having a point in common. Let **w** be a point common to all members of a nonempty collection **x** of connected subspaces. Let **y ∈ x** and let **z** be an open and closed set. If **w ∈ z**, then **y ⊂ z**.

Lemma.

```
In[2]:= Map[not, SubstTest[and, implies[and[p1, p3], p7],
           implies[and[p1, p2], p8], implies[and[p0, p1], p9],
           implies[and[p6, p7], p10], (*implies[and[p4,p5,p8,p9,p10],p11],*)
           not[implies[and[p0, p1, p2, p3, p4, p5, p6], p11]],
           {p0 → subclass[U[x], U[t]], p1 → member[y, x], p2 -> subclass[
              image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], CONNECTED],
            p3 → member[w, A[x]], p4 → member[z, t], p5 → member[dif[U[t], z], t],
            p6 → member[w, z], p7 → member[w, y], p8 → member[image[IMAGE[id[y]], t], CONNECTED],
            p9 → subclass[y, U[t]], p10 → not[disjoint[y, z]], p11 → subclass[y, z]}]] // Reverse

Out[2]= or[not[member[w, z]], not[member[w, A[x]]], not[member[y, x]],
          not[member[z, t]], not[member[intersection[complement[z], U[t]], t]],
          not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
            CONNECTED]], not[subclass[U[x], U[t]]], subclass[y, z]] == True

In[3]:= (% /. {t → t_, w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

To eliminate the variable **y** it helps to clear the **simplify** and **cond** flags.

```
In[4]:= simplify = False; cond = False;
```

Lemma. (Eliminate the variable **y**.)

```
In[5]:= Map[equal[V, domain[#]] &,
         SubstTest[reify, y, case[implies[and[member[w, z], member[w, A[x]], member[y, x],
             member[z, s], subclass[u, v], subclass[U[x], U[t]]], subclass[y, z]]],
          {s → intersection[t, image[RC[U[t]], t]], u ->
            image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], v -> CONNECTED}]]

Out[5]= or[not[member[t, V]], not[member[w, z]], not[member[w, A[x]]],
          not[member[z, t]], not[member[intersection[complement[z], U[t]], t]],
          not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
            CONNECTED]], not[subclass[U[x], U[t]]], subclass[U[x], z]] == True

In[6]:= (% /. {t → t_, w → w_, x → x_, z → z_}) /. Equal → SetDelayed
```

Lemma. Replace **z** with its relative complement in **U[t]**.

```
In[7]:= SubstTest[or, not[member[t, V]], not[member[w, s]], not[member[w, A[x]]],
           not[member[s, t]], not[member[intersection[complement[s], U[t]], t]], not[subclass[
             image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], CONNECTED]],
           not[subclass[U[x], U[t]]], subclass[U[x], s], s → dif[U[t], z]] // Reverse // MapNotNot

Out[7]= or[equal[0, intersection[z, U[x]]], member[w, z], not[member[t, V]],
          not[member[w, A[x]]], not[member[w, U[t]]], not[member[intersection[z, U[t]], t]],
          not[member[intersection[complement[z], U[t]], t]],
          not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
            CONNECTED]], not[subclass[U[x], U[t]]]] == True

In[8]:= (% /. {t → t_, w → w_, x → x_, z → z_}) /. Equal → SetDelayed
```

Lemma. (Simplify the previous result.)

```
In[9]:= SubstTest[implies, equal[s, intersection[z, U[t]]], or[equal[0, intersection[z, U[x]]],
          member[w, z], not[member[t, V]], not[member[w, A[x]]], not[member[w, U[t]]],
          not[member[s, t]], not[member[intersection[complement[z], U[t]], t]],
          not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
            CONNECTED]], not[subclass[U[x], U[t]]]], s → z] // Reverse

Out[9]= or[equal[0, intersection[z, U[x]]], member[w, z], not[member[t, V]],
          not[member[w, A[x]]], not[member[w, U[t]]], not[member[z, t]],
          not[member[intersection[complement[z], U[t]], t]], not[subclass[z, U[t]]],
          not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
            CONNECTED]], not[subclass[U[x], U[t]]]] == True

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

Theorem. Combine the two cases.

```
In[11]:= Map[not,
           SubstTest[and, implies[and[not[p0], not[p1]], p2], implies[and[p0, not[p1]], p3],
            not[implies[not[p1], or[p2, p3]]], {p0 → member[w, z], p1 -> or[not[member[w, A[x]]],
               not[member[w, U[t]]], not[member[z, t]], not[member[t, V]],
               not[member[intersection[complement[z], U[t]], t]], not[subclass[z, U[t]]],
               not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
                 CONNECTED]], not[subclass[U[x], U[t]]]],
             p2 -> equal[0, intersection[U[x], z]], p3 → subclass[U[x], z]}]] // Reverse

Out[11]= or[equal[0, intersection[z, U[x]]], not[member[t, V]],
           not[member[w, A[x]]], not[member[w, U[t]]], not[member[z, t]],
           not[member[intersection[complement[z], U[t]], t]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
             CONNECTED]], not[subclass[U[x], U[t]]], subclass[U[x], z]] == True

In[12]:= (% /. {t → t_, w → w_, x → x_, z → z_}) /. Equal → SetDelayed
```

Lemma. Eliminate the redundant literal $\mathbf{w \in U[t]}$.

```
In[13]:= Map[not, SubstTest[and, implies[and[p0, p1], p2], implies[and[p1, p2], p3],
            implies[and[p1, p3], or[p4, p5]], implies[not[p0], or[p4, p5]],
            not[implies[p1, or[p4, p5]]], {p0 → not[empty[x]], p1 →
              and[member[t, V], member[w, A[x]], member[z, intersection[t, image[RC[U[t]], t]]],
               subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
                CONNECTED], subclass[U[x], U[t]]],
             p2 → member[w, U[x]], p3 → member[w, U[t]], p4 -> subclass[U[x], z],
             p5 → disjoint[U[x], z]}]] // Reverse

Out[13]= or[equal[0, intersection[z, U[x]]], not[member[t, V]], not[member[w, A[x]]],
           not[member[z, t]], not[member[intersection[complement[z], U[t]], t]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
             CONNECTED]], not[subclass[U[x], U[t]]], subclass[U[x], z]] == True

In[14]:= (% /. {t → t_, w → w_, x → x_, z → z_}) /. Equal → SetDelayed
```

Lemma. Eliminate the variable $\mathbf{w}$.

```
In[15]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, w, case[or[equal[0, intersection[z, U[x]]], not[member[t, V]],
             not[member[w, A[x]]], not[member[z, s]], not[subclass[u, v]],
             not[subclass[U[x], U[t]]], subclass[U[x], z]]],
           {s -> intersection[t, image[RC[U[t]], t]], u -> image[
              VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], v -> CONNECTED}]]

Out[15]= or[equal[0, A[x]], equal[0, intersection[z, U[x]]], not[member[t, V]],
           not[member[z, t]], not[member[intersection[complement[z], U[t]], t]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
             CONNECTED]], not[subclass[U[x], U[t]]], subclass[U[x], z]] == True

In[16]:= (% /. {t → t_, x → x_, z → z_}) /. Equal → SetDelayed
```

At this point it is useful to specialize to the case that **U[x] = U[t]**, that is, the union of the collection of connected subpaces is the whole space. This limitation will be removed later on.

Theorem. Specialization to the case **U[x] = U[t]**.

```
In[17]:= Map[not, SubstTest[and, implies[and[p0, p1], p2],
            implies[and[p1, p2], or[p3, p4]], implies[and[p0, p1, p3], p5],
            implies[and[p0, p1, p4], p6], not[implies[and[p0, p1], or[p5, p6]]],
            {p0 -> equal[U[x], U[t]], p1 → and[member[t, V], not[empty[A[x]]],
               subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
                CONNECTED], member[z, intersection[t, image[RC[U[t]], t]]]],
             p2 -> subclass[U[x], U[t]], p3 → subclass[U[x], z], p4 -> disjoint[U[x], z],
             p5 → equal[z, U[t]], p6 → equal[0, z]}]] // Reverse

Out[17]= or[equal[0, z], equal[0, A[x]], equal[z, U[t]],
           not[equal[U[t], U[x]]], not[member[t, V]], not[member[z, t]],
           not[member[intersection[complement[z], U[t]], t]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
             CONNECTED]]] == True

In[18]:= (% /. {t → t_, x → x_, z → z_}) /. Equal → SetDelayed
```

Theorem. (Eliminate the variable **z**.)

```
In[19]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, z, case[or[equal[0, z], equal[0, A[x]], equal[z, U[t]],
             not[equal[U[t], U[x]]], not[member[t, V]], not[member[z, s]], not[subclass[u, v]]]],
           {s -> intersection[t, image[RC[U[t]], t]], u ->
             image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], v → CONNECTED}]]

Out[19]= or[equal[0, A[x]], not[equal[U[t], U[x]]], not[member[t, V]], not[subclass[
             image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], CONNECTED]],
           subclass[intersection[t, image[RC[U[t]], t]], set[0, U[t]]]] == True

In[20]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

The limitation **U[x] = U[t]** will now be removed by replacing **t** with the relative topology for the subspace **U[x]**.

Lemma. A sethood rule.

```
In[21]:= SubstTest[member, U[y], V, y -> image[IMAGE[id[x]], t]]

Out[21]= member[image[IMAGE[id[x]], t], V] == member[intersection[x, U[t]], V]

In[22]:= member[image[IMAGE[id[x_]], t_], V] := member[intersection[x, U[t]], V]
```

Lemma.

```
In[23]:= SubstTest[or, equal[0, A[x]], not[equal[U[s], U[x]]], not[member[s, V]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, s]], inverse[FIRST]]], x],
             CONNECTED]], subclass[intersection[s, image[RC[U[s]], s]],
            set[0, U[s]]], s → image[IMAGE[id[U[x]]], t]] // Reverse

Out[23]= or[equal[0, A[x]], not[member[intersection[U[t], U[x]], V]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
             CONNECTED]], not[subclass[U[x], U[t]]],
           subclass[intersection[image[IMAGE[id[U[x]]], t], image[RC[intersection[U[t], U[x]]],
              image[IMAGE[id[U[x]]], t]]], set[0, intersection[U[t], U[x]]]]] == True

In[24]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma. Simplification of the preceding lemma.

```
In[25]:= Map[implies[member[x, y], #] &, SubstTest[implies, equal[w, intersection[U[t], U[x]]],
            or[equal[0, A[x]], not[member[w, V]], not[subclass[
              image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], CONNECTED]],
             not[subclass[U[x], U[t]]], subclass[intersection[image[IMAGE[id[U[x]]], t],
              image[RC[w], image[IMAGE[id[U[x]]], t]]], set[0, w]]], w → U[x]]] // Reverse

Out[25]= or[equal[0, A[x]], not[member[x, y]], not[subclass[
             image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x], CONNECTED]],
           not[subclass[U[x], U[t]]], subclass[intersection[image[IMAGE[id[U[x]]], t],
             image[RC[U[x]], image[IMAGE[id[U[x]]], t]]], set[0, U[x]]]] == True

In[26]:= or[equal[0, A[x_]], not[member[x_, y_]], not[subclass[
             image[VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]], x_], CONNECTED]],
           not[subclass[U[x_], U[t_]]], subclass[intersection[image[IMAGE[id[U[x_]]], t_],
             image[RC[U[x_]], image[IMAGE[id[U[x_]]], t_]]], set[0, U[x_]]]] := True
```

Theorem. The union of a collection of connected subspaces having a point in common is connected.

```
In[27]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p2, p3], p4],
           implies[and[p1, p2, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 → and[member[0, t], member[U[t], t]], p2 → and[not[equal[0, A[x]]],
              subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
               CONNECTED], subclass[U[x], U[t]]], p3 → member[x, V],
            p4 -> subclass[intersection[image[IMAGE[id[U[x]]], t],
               image[RC[U[x]], image[IMAGE[id[U[x]]], t]]], set[0, U[x]]],
            p5 → member[image[IMAGE[id[U[x]]], t], CONNECTED]}]] // Reverse

Out[27]= or[equal[0, A[x]], member[image[IMAGE[id[U[x]]], t], CONNECTED],
           not[member[0, t]], not[member[U[t], t]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], x],
             CONNECTED]], not[subclass[U[x], U[t]]]] == True

In[28]:= or[equal[0, A[x_]], member[image[IMAGE[id[U[x_]]], t_], CONNECTED],
           not[member[0, t_]], not[member[U[t_], t_]],
           not[subclass[image[VERTSECT[composite[CAP, id[cart[V, t_]], inverse[FIRST]]], x_],
             CONNECTED]], not[subclass[U[x_], U[t_]]]] := True
```

Lemma. The union of two non-disjoint connected subspaces is connected.

```
In[29]:= SubstTest[or, equal[0, A[w]], member[image[IMAGE[id[U[w]]], t], CONNECTED],
           not[member[0, t]], not[member[U[t], t]], not[subclass[
             image[VERTSECT[composite[CAP, id[cart[V, t]], inverse[FIRST]]], w], CONNECTED]],
           not[subclass[U[w], U[t]]], w → set[setpart[x], setpart[y]]] // Reverse

Out[29]= or[equal[0, intersection[setpart[x], setpart[y]]],
           member[image[IMAGE[id[union[setpart[x], setpart[y]]], t], CONNECTED],
           not[member[0, t]], not[member[image[IMAGE[id[setpart[x]]], t], CONNECTED]],
           not[member[image[IMAGE[id[setpart[y]]], t], CONNECTED]], not[member[U[t], t]],
           not[subclass[setpart[x], U[t]]], not[subclass[setpart[y], U[t]]]] == True

In[30]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

The **setpart** wrappers can be removed.

Lemma.

```
In[31]:= SubstTest[implies, and[equal[x, setpart[u]], equal[y, setpart[v]]],
           or[equal[0, intersection[x, y]], member[image[IMAGE[id[union[x, y]]], t], CONNECTED],
            not[member[0, t]], not[member[image[IMAGE[id[x]], t], CONNECTED]],
            not[member[image[IMAGE[id[y]], t], CONNECTED]], not[member[U[t], t]],
            not[subclass[x, U[t]]], not[subclass[y, U[t]]]], {u → x, v → y}] // Reverse

Out[31]= or[equal[0, intersection[x, y]],
           member[image[IMAGE[id[union[x, y]]], t], CONNECTED], not[member[0, t]],
           not[member[x, V]], not[member[y, V]], not[member[image[IMAGE[id[x]], t], CONNECTED]],
           not[member[image[IMAGE[id[y]], t], CONNECTED]], not[member[U[t], t]],
           not[subclass[x, U[t]]], not[subclass[y, U[t]]]] == True

In[32]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Since **U[t] ∈ t** implies that **t** is a set, the inclusions **x ⊂ U[t]** and **y ⊂ U[t]** imply that **x** and **y** are sets. This one does not need explicit sethood literals for **x** and **y**.

Theorem. The union of non-disjoint connected subspaces is connected.

```
In[38]:= Map[not, SubstTest[and, implies[and[p1, p2], p4],
           implies[and[p1, p2], p5], implies[and[p1, p2, p3, p4, p5], p6],
           not[implies[and[p1, p2, p3], p6]], {p1 → and[member[0, t], member[U[t], t]],
            p2 → and[subclass[x, U[t]], subclass[y, U[t]]], p3 → and[member[
               image[IMAGE[id[x]], t], CONNECTED], member[image[IMAGE[id[y]], t], CONNECTED]],
            p4 → member[x, V], p5 → member[y, V], p6 -> or[equal[0, intersection[x, y]],
              member[image[IMAGE[id[union[x, y]]], t], CONNECTED]}]] // Reverse

Out[38]= or[equal[0, intersection[x, y]], member[image[IMAGE[id[union[x, y]]], t], CONNECTED],
           not[member[0, t]], not[member[image[IMAGE[id[x]], t], CONNECTED]],
           not[member[image[IMAGE[id[y]], t], CONNECTED]], not[member[U[t], t]],
           not[subclass[x, U[t]]], not[subclass[y, U[t]]]] == True

In[40]:= or[equal[0, intersection[x_, y_]],
           member[image[IMAGE[id[union[x_, y_]]], t_], CONNECTED],
           not[member[0, t_]], not[member[image[IMAGE[id[x_]], t_], CONNECTED]],
           not[member[image[IMAGE[id[y_]], t_], CONNECTED]], not[member[U[t_], t_]],
           not[subclass[x_, U[t_]]], not[subclass[y_, U[t_]]]] := True
```