a subspace of a T2 space is T2

Johan G. F. Belinfante
2011 April 4

```
In[1]:= SetDirectory["l:"]; << goedel.11apr03a

        :Package Title: goedel.11apr03a               2011 April 3 at 5:55 p.m.

        It is now:  2011 Apr 4 at 14:40

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

Any subspace of a Hausdorff space is a Hausdorff space. In this notebook the **T2** separation condition is considered independently of topology. If a collection **t** of sets satisfies the **T2** separation condition, and if **x** is any class, then the collection of sets obtained by intersecting each member of **t** with **x** also satisfies the **T2** condition.

derivation

Lemma.

```
In[2]:= SubstTest[implies, subclass[u, v], subclass[image[z, u], image[z, v]],
          {u -> composite[id[t], DISJOINT, id[t]], v -> DISJOINT,
           z → cross[IMAGE[id[x]], composite[IMAGE[id[x]]]]}] // Reverse

Out[2]= subclass[composite[IMAGE[id[x]], id[t],
           DISJOINT, id[t], inverse[IMAGE[id[x]]]], DISJOINT] == True

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

Lemma.

```
In[4]:= SubstTest[implies, subclass[u, v],
          subclass[image[z, u], image[z, v]], {u -> composite[id[U[t]], Di, id[U[t]]], v ->
            composite[inverse[E], id[t], DISJOINT, id[t], E], z → cross[id[x], id[x]]}] // Reverse

Out[4]= or[not[subclass[composite[id[U[t]], Di, id[U[t]]],
            composite[inverse[E], id[t], DISJOINT, id[t], E]]],
          subclass[composite[id[intersection[x, U[t]]], Di, id[intersection[x, U[t]]]],
           composite[inverse[E], IMAGE[id[x]], id[t],
            DISJOINT, id[t], inverse[IMAGE[id[x]]], E]]] == True
```

```
In[5]:= (% /. {x → x_, t → t_}) /. Equal → SetDelayed
```

```
In[6]:= SubstTest[implies, and[member[z, V], subclass[composite[id[U[z]], Di, id[U[z]]],
            composite[inverse[E], id[z], DISJOINT, id[z], E]]],
          member[z, T2], z -> image[IMAGE[id[x]], t]] // Reverse
```

```
Out[6]= or[member[image[IMAGE[id[x]], t], T2], not[member[intersection[x, U[t]], V]],
          not[subclass[composite[id[intersection[x, U[t]]], Di, id[intersection[x, U[t]]]],
            composite[inverse[E], id[image[IMAGE[id[x]], t]],
             DISJOINT, id[image[IMAGE[id[x]], t]], E]]]] == True
```

```
In[7]:= (% /. {x → x_, t → t_}) /. Equal → SetDelayed
```

Main Theorem. If **t ∈ T2**, then **image[IMAGE[id[x]], t] ∈ T2**.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
           implies[p1, p5], implies[and[p4, p5], p6], not[implies[p1, p6]],
           {p1 → member[t, T2], p2 -> subclass[composite[id[U[t]], Di, id[U[t]]],
              composite[inverse[E], id[t], DISJOINT, id[t], E]], p3 -> subclass[
              composite[id[intersection[x, U[t]]], Di, id[intersection[x, U[t]]]], composite[
               inverse[E], IMAGE[id[x]], id[t], DISJOINT, id[t], inverse[IMAGE[id[x]]], E]],
            p4 -> subclass[composite[id[intersection[x, U[t]]], Di, id[intersection[x, U[t]]]],
              composite[inverse[E], id[image[IMAGE[id[x]], t]], DISJOINT,
               id[image[IMAGE[id[x]], t]], E]], p5 -> member[intersection[x, U[t]], V],
            p6 -> member[image[IMAGE[id[x]], t], T2]}]] // Reverse
```

```
Out[8]= or[member[image[IMAGE[id[x]], t], T2], not[member[t, T2]]] == True
```

```
In[9]:= or[member[image[IMAGE[id[x_]], t_], T2], not[member[t_, T2]]] := True
```

Corollary. The class **T2** is invariant under **IMAGE[IMAGE[id[t]]**.

```
In[10]:= Map[equal[V, #] &,
          complement[dif[T2, image[inverse[IMAGE[IMAGE[id[x]]]], T2]]] // Normality]
```

```
Out[10]= subclass[image[IMAGE[IMAGE[id[x]]], T2], T2] == True
```

```
In[11]:= (% /. x → x_) /. Equal → SetDelayed
```

The above inclusion can be sharpened to an equation.

Theorem.

```
In[17]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → IMAGE[IMAGE[id[y]]], u → intersection[x, P[P[y]]], v → x}] // Reverse
```

```
Out[17]= subclass[intersection[x, P[P[y]]], image[IMAGE[IMAGE[id[y]]], x]] == True
```

```
In[18]:= subclass[intersection[x_, P[P[y_]]], image[IMAGE[IMAGE[id[y_]]], x_]] := True
```

Corollary.

```
In[19]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[IMAGE[IMAGE[id[x]]], T2], v → intersection[T2, P[P[x]]]}]

Out[19]= equal[image[IMAGE[IMAGE[id[x]]], T2], intersection[T2, P[P[x]]]] == True

In[21]:= image[IMAGE[IMAGE[id[x_]]], T2] := intersection[T2, P[P[x]]]
```

A variable-free formulation is possible.

Theorem.

```
In[22]:= SubstTest[reify, x, image[IMAGE[IMAGE[id[x]]], t], t → T2]

Out[22]= composite[IMAGE[CAP], CART, id[cart[V, T2]], inverse[FIRST], SINGLETON] ==
          composite[id[T2], inverse[S], POWER]

In[23]:= composite[IMAGE[CAP], CART, id[cart[V, T2]], inverse[FIRST], SINGLETON] :=
          composite[id[T2], inverse[S], POWER]
```

Corollary.

```
In[25]:= ImageComp[composite[IMAGE[CAP], CART, id[cart[V, T2]], inverse[FIRST]], SINGLETON, V] //
          Reverse

Out[25]= image[IMAGE[CAP], image[CART, cart[range[SINGLETON], T2]]] == T2

In[26]:= image[IMAGE[CAP], image[CART, cart[range[SINGLETON], T2]]] := T2
```