T1 separation condition

Johan G. F. Belinfante
2003 June 23

```
In[1]:= << goedel52.s13; << tools.m

        :Package Title: goedel52.s13       2003 June 23 at 4:40 p.m.

        It is now:  2003 Jun 25 at 10:50

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 June 13

        weightlimit = 40
```

■ summary

It is shown that the T2 separation condition implies the T1 separation condition. Although the intended application is to topology, neither of the two topology axioms are assumed. The T2 condition was studied earlier; recall that **T2** is defined as the class of all collections of sets for which the T2 separation condition holds:

```
In[2]:= member[t, T2]

Out[2]= and[member[t, V], subclass[composite[id[U[t]], Di, id[U[t]]],
          composite[inverse[∈], id[t], DISJOINT, id[t], ∈]]]
```

A Hausdorff topology is a topology that satisfies the T2 separation condition. Note that the variable **t** occurs four times on the right side of the membership rule for **T2**.

■ T1 separation condition

Although the T1 separation condition is introduced here without regard to topology, it is easiest to understand the condition if one thinks about the case of a topology. If **t** is a topology satisfying the T1 condition, and **x** is a point of the underlying topological space **U[t]**, then for every point **y** distinct from **x**, there is an open set **z** which holds **x** and does not hold **y**. It is intutively clear that it does not matter whether or not the point **y** belongs to the topological space, so for simplicity no condition is placed on **y** other than that it is a point different from **x**. (A formal demonstration that one's intuition is correct is given below.) The T1 separation condition thus amounts to this statement:

```
In[3]:= assert[forall[x, y, implies[and[member[x, U[t]], not[equal[x, y]]],
                    exists[z, and[member[x, z], member[z, t], not[member[y, z]]]]]]]

Out[3]= subclass[composite[Di, id[U[t]]], composite[complement[inverse[∈]], id[t], ∈]]
```

The class of all collections of sets that satisfy this condition will be called **T1**, whether these collections satisfy the topology axioms or not. Note that the variable **t** appears twice on the right side of this membership rule

```
In[4]:= member[t_, T1] := and[member[t, V],
          subclass[composite[Di, id[U[t]]], composite[complement[inverse[E]], id[t], E]]]
```

Any power set satisfies this condition, so the class **T1** is a proper class.

```
In[5]:= Map[equal[V, #] &, image[inverse[POWER], T1] // Normality]

Out[5]= subclass[range[POWER], T1] == True

In[6]:= subclass[range[POWER], T1] := True

In[7]:= Map[not, SubstTest[implies, and[subclass[u, v], member[v, V]],
          member[u, V], {u -> range[POWER], v -> T1}]]

Out[7]= member[T1, V] == False

In[8]:= member[T1, V] := False
```

More generally:

```
In[9]:= member[T1, x] // AssertTest

Out[9]= member[T1, x] == False

In[10]:= member[T1, x_] := False
```

■ Normality considerations

When the **Normality** test is applied to **T1** an expression is obtaiined involving an upper bound construction **UB** that can be simplified:

```
In[11]:= UB[union[composite[inverse[FIRST], complement[inverse[E]]],
            composite[inverse[SECOND], inverse[E]]]] // VSNormality

Out[11]= UB[union[composite[inverse[FIRST], complement[inverse[∈]]],
            composite[inverse[SECOND], inverse[∈]]]] ==
          union[cart[singleton[0], V], composite[id[cart[V, V]], inverse[SINGLETON], LB[INVAR]]]

In[12]:= UB[union[composite[inverse[FIRST], complement[inverse[E]]],
            composite[inverse[SECOND], inverse[E]]]] :=
          union[cart[singleton[0], V], composite[id[cart[V, V]], inverse[SINGLETON], LB[INVAR]]]
```

Repeating the **Normality** test with this rule in place yields a fairly compact formula for the class **T1**.

```
In[13]:= Map[complement, T1 // Normality // Reverse]

Out[13]= fix[composite[inverse[LB[INVAR]], SINGLETON,
            id[Di], inverse[FIRST], inverse[∈], BIGCUP]] == complement[T1]

In[14]:= fix[composite[inverse[LB[INVAR]], SINGLETON,
            id[Di], inverse[FIRST], inverse[E], BIGCUP]] := complement[T1]
```

The following is a variant of this:

```
In[15]:= fix[composite[inverse[BIGCUP], E, FIRST, id[Di], inverse[SINGLETON], LB[INVAR]]] //
          InvertFixTest

Out[15]= fix[composite[inverse[BIGCUP], ∈, FIRST, id[Di], inverse[SINGLETON], LB[INVAR]]] ==
          complement[T1]
```

```
In[16]:= fix[composite[inverse[BIGCUP], E, FIRST, id[Di], inverse[SINGLETON], LB[INVAR]]] :=
          complement[T1]
```

■ a lemma

To show that the **T2** condition implies the **T1** condition, one needs the lemma derived in this section. For the **T2** condition it is important that both of the points **x** and **y** belong to the topological space **U[t]**. If one adds to the T1 condition the hypothesis that **y** belongs to **U[t]**, one obtains the following variant of the T1 condition:

```
In[17]:= assert[forall[x, y, implies[and[member[x, U[t]], member[y, U[t]], not[equal[x, y]]],
                    exists[z, and[member[x, z], member[z, t], not[member[y, z]]]]]]]

Out[17]= subclass[composite[id[U[t]], Di, id[U[t]]],
          composite[complement[inverse[∈]], id[t], ∈]]
```

Note that the variable **t** occurs three times in this formula, but only twice in the T1 condition. To show that the variant condition is really the same as the original T1 condition, the following identity is used:

```
In[18]:= union[cart[U[t], complement[U[t]]], composite[id[U[t]], Di, id[U[t]]]]

Out[18]= composite[Di, id[U[t]]]
```

The following key ideas address the **cart[U[t],complement[U[t]]]** term in this equation.

```
In[19]:= subclass[
           class[pair[pair[x, y], z], and[member[x, z], member[z, t], not[member[y, U[t]]]]],
         class[pair[pair[x, y], z], and[member[x, z], member[z, t], not[member[y, z]]]] //
          AssertTest

Out[19]= subclass[composite[id[t], ∈, FIRST, id[cart[V, complement[U[t]]]]],
           composite[complement[INVAR], SINGLETON]] == True

In[20]:= subclass[composite[id[t_], E, FIRST, id[cart[V, complement[U[t_]]]]],
           composite[complement[INVAR], SINGLETON]] := True

In[21]:= SubstTest[implies, subclass[u, v], subclass[domain[u], domain[v]],
             {u -> composite[id[t], E, FIRST, id[cart[V, complement[U[t]]]]],
           v -> composite[id[t], complement[INVAR], SINGLETON]}]

Out[21]= subclass[cart[U[t], complement[U[t]]],
           composite[complement[inverse[∈]], id[t], ∈]] == True

In[22]:= subclass[cart[U[t_], complement[U[t_]]],
           composite[complement[inverse[E]], id[t_], E]] := True
```

The whole argument that the variant T1 condition is equivalent to the original T1 condition can now be presented compactly:

```
In[23]:= Map[complement[class[t, #]] &, SubstTest[subclass, union[x, y], z,
             {x -> cart[U[t], complement[U[t]]], y -> composite[id[U[t]], Di, id[U[t]]],
             z -> composite[complement[inverse[E]], id[t], E]}]] // Reverse

Out[23]= fix[composite[inverse[LB[INVAR]], inverse[∈],
           IMAGE[SINGLETON], IMAGE[id[Di]], CART, DUP, BIGCUP]] == complement[T1]

In[24]:= fix[composite[inverse[LB[INVAR]], inverse[E],
           IMAGE[SINGLETON], IMAGE[id[Di]], CART, DUP, BIGCUP]] := complement[T1]
```

■ T2 implies T1

To complete the proof that T2 implies T1, one needs to remove one of the **id[t]** factors in the T2 condition:

```
In[25]:= member[t, T2]

Out[25]= and[member[t, V], subclass[composite[id[U[t]], Di, id[U[t]]],
           composite[inverse[e], id[t], DISJOINT, id[t], e]]]
```

The removal of the factor yields this inequality:

```
In[26]:= SubstTest[subclass, composite[x, id[t], y], composite[x, y],
             {x -> inverse[E], y -> composite[DISJOINT, id[t], E]}]

Out[26]= subclass[composite[inverse[e], id[t], DISJOINT, id[t], e],
           composite[complement[inverse[e]], id[t], e]] == True

In[27]:= subclass[composite[inverse[E], id[t_], DISJOINT, id[t_], E],
           composite[complement[inverse[E]], id[t_], E]] := True
```

The final step translates an implication to an inclusion:

```
In[28]:= Map[equal[V, #] &, SubstTest[class, t, implies[p[t], q[t]],
             {p[t] -> subclass[composite[id[U[t]], Di, id[U[t]]],
               composite[inverse[E], id[t], DISJOINT, id[t], E]],
                 q[t] -> subclass[composite[id[U[t]], Di, id[U[t]]],
               composite[complement[inverse[E]], id[t], E]]}]] // Reverse

Out[28]= subclass[T2, T1] == True

In[29]:= subclass[T2, T1] := True
```