COARSER

Johan G. F. Belinfante
2003 July 4

```
In[1]:= << goedel52.s38; << tools.m

        :Package Title: goedel52.s38       2003 July 4 at 8:45 a.m.

        It is now:  2003 Jul 6 at 21:6

        Loading Simplification Rules

        TOOLS.M                      Revised 2003 July 4

        weightlimit = 40
```

■ summary

A topology **x** is coarser than a topology **y** if both are topologies for the same topological space, and **x** is contained in **y**. This topological concept is generalized here to arbitrary collections of sets. The relation **COARSER** is defined as the class of all **pair[x,y]** such that **subclass[x,y]** and **U[x]=U[y]**. Elementary properties this relation are deduced. A concise statement is derived which expresses the observation that a collection of sets satisfies the T1 separation condition if a coarser collection does.

The **GOEDEL** program contains this description of the relation:

```
In[2]:= ?COARSER

        pair[x,y] belongs to COARSER if x is a subset of y and U[x] = U[y]
```

■ reference: John L. Kelley, General Topology, Van Nostrand Co., Inc., Princeton, N.J., 1955.

The definition of coarser is given on page 38 of Kelley's book. There is an exercise about T1 spaces on page 56.

■ membership condition for pairs

The defining membership relation for **COARSER** is wrapped in **class**. From this wrapped definition it is easy to derive the following useful special membership relation for pairs:

```
In[3]:= equiv[member[pair[x, y], COARSER],
          and[member[y, V], subclass[x, y], equal[U[x], U[y]]]] //
         assert

Out[3]= True
```

This special case will be added in unwrapped form:

```
In[4]:= member[pair[x_, y_], COARSER] := and[member[y, V], subclass[x, y], equal[U[x], U[y]]]
```

■ normalization

The **Renormality** test yields an equation that could be used to define the relation **COARSER**. For work in **Otter**, this formula could serve as a definition from which the membership rule would have to be deduced.

```
In[5]:= COARSER // Renormality // Reverse

Out[5]= intersection[S, composite[inverse[S], POWER, BIGCUP]] == COARSER

In[6]:= intersection[S, composite[inverse[S], POWER, BIGCUP]] := COARSER
```

An immediate corollary:

```
In[7]:= SubstTest[subclass, intersection[x, y], x,
         {x -> S, y -> composite[inverse[S], POWER, BIGCUP]}]

Out[7]= subclass[COARSER, S] == True
```

The same fact can also be deduced another way:

```
In[8]:= Map[equal[0, #]&, dif[COARSER, S] // Renormality]

Out[8]= subclass[COARSER, S] == True

In[9]:= subclass[COARSER, S] := True
```

This implies that **COARSER** is a relation:

```
In[10]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> COARSER, v -> S, w -> cart[V, V]}]

Out[10]= subclass[COARSER, cart[V, V]] == True

In[11]:= subclass[COARSER, cart[V, V]] := True
```

The associative law for **intersection** yields the following equivalent formulation. This is a useful simplification rule:

```
In[12]:= AssInt[cart[V, V], S, composite[inverse[S], POWER, BIGCUP]]

Out[12]= composite[Id, COARSER] == COARSER

In[13]:= composite[Id, COARSER] := COARSER
```

A further consequence is that **inverse[COARSER]** is a thin relation:

```
In[14]:= SubstTest[implies, and[subclass[x, y], thin[y]], thin[x],
          {x -> inverse[COARSER], y -> inverse[S]}]

Out[14]= equal[V, domain[VERTSECT[inverse[COARSER]]]] == True

In[15]:= domain[VERTSECT[inverse[COARSER]]] := V
```

■ further simplifications

Normality yields the following simplification rule:

```
In[16]:= COARSER // Normality // Reverse

Out[16]= intersection[COARSER, composite[inverse[BIGCUP], inverse[POWER], S]] == COARSER

In[17]:= intersection[COARSER, composite[inverse[BIGCUP], inverse[POWER], S]] := COARSER
```

From this follows the equivalence of two different descriptions of the relation **COARSER**.

```
In[18]:= class[pair[x, y], and[subclass[x, y], subclass[y, P[U[x]]]]]

Out[18]= COARSER

In[19]:= class[pair[x, y], and[subclass[x, y], equal[U[x], U[y]]]]

Out[19]= COARSER
```

■ domain and range

The domain is easily deduced.

```
In[20]:= domain[COARSER] // Renormality

Out[20]= domain[COARSER] == V

In[21]:= domain[COARSER] := V
```

The relation **COARSER** is reflexive:

```
In[22]:= fix[COARSER] // Normality

Out[22]= fix[COARSER] == V

In[23]:= fix[COARSER] := V
```

A formula for the range follows from this result:

```
In[24]:= SubstTest[subclass, fix[x], range[x], x -> COARSER]
Out[24]= subclass[V, range[COARSER]] == True

In[25]:= subclass[V, range[COARSER]] := True

In[26]:= equal[range[COARSER], V] // AssertTest
Out[26]= equal[V, range[COARSER]] == True

In[27]:= range[COARSER] := V
```

■ transitivity

```
In[28]:= SubstTest[implies,
           and[subclass[u, v], subclass[x, y]], subclass[composite[u, x], composite[v, y]],
           {u -> COARSER, v -> S, x -> COARSER, y -> S}]
Out[28]= subclass[composite[COARSER, COARSER], S] == True

In[29]:= subclass[composite[COARSER, COARSER], S] := True

In[30]:= SubstTest[subclass, intersection[x, y], y,
           {x -> S, y -> composite[inverse[S], POWER, BIGCUP]}]
Out[30]= subclass[COARSER, composite[inverse[S], POWER, BIGCUP]] == True

In[31]:= subclass[COARSER, composite[inverse[S], POWER, BIGCUP]] := True
```

Lemma:

```
In[32]:= composite[BIGCUP, COARSER] // RelnRenormality
Out[32]= composite[BIGCUP, COARSER] == BIGCUP

In[33]:= composite[BIGCUP, COARSER] := BIGCUP

In[34]:= SubstTest[implies,
           and[subclass[u, v], subclass[x, y]], subclass[composite[u, x], composite[v, y]],
           {u -> COARSER, x -> COARSER, y -> COARSER, v -> composite[inverse[S], POWER, BIGCUP]}]
Out[34]= subclass[composite[COARSER, COARSER], composite[inverse[S], POWER, BIGCUP]] == True

In[35]:= subclass[composite[COARSER, COARSER], composite[inverse[S], POWER, BIGCUP]] := True

In[36]:= SubstTest[subclass, u, intersection[v, w],
           {u -> composite[COARSER, COARSER], v -> S, w -> composite[inverse[S], POWER, BIGCUP]}]
Out[36]= subclass[composite[COARSER, COARSER], COARSER] == True

In[37]:= subclass[composite[COARSER, COARSER], COARSER] := True
```

```
In[38]:= TRANSITIVE[COARSER] // AssertTest

Out[38]= TRANSITIVE[COARSER] == True

In[39]:= TRANSITIVE[COARSER] := True
```

The reverse inclusion also holds:

```
In[40]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
          {u -> Id, v -> COARSER, w -> COARSER}]

Out[40]= subclass[COARSER, composite[COARSER, COARSER]] == True

In[41]:= subclass[COARSER, composite[COARSER, COARSER]] := True
```

The two inclusions can be combined into an equation:

```
In[42]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> composite[COARSER, COARSER], v -> COARSER}]

Out[42]= True == equal[COARSER, composite[COARSER, COARSER]]

In[43]:= composite[COARSER, COARSER] := COARSER
```

The antisymmetric law also holds:

```
In[44]:= intersection[inverse[COARSER], COARSER] // RelnNormality // InvertFix

Out[44]= intersection[COARSER, inverse[COARSER]] == Id

In[45]:= intersection[COARSER, inverse[COARSER]] := Id
```

It follows that **COARSER** is a partial ordering.

```
In[46]:= PARTIALORDER[COARSER]

Out[46]= True
```

■ an application to the T1 separation condition

The following general result is needed:

```
In[47]:= SubstTest[implies, equal[u, v], equal[composite[w, u, z], composite[w, v, z]],
          {u -> id[x], v -> id[y]}]

Out[47]= or[equal[composite[w, id[x], z], composite[w, id[y], z]], not[equal[x, y]]] == True

In[48]:= or[equal[composite[w___, id[x_], z___],
            composite[w___, id[y_], z___]], not[equal[x_, y_]]] :=
          True
```

The reasoning needed for the application is:

```
In[49]:= Map[not, SubstTest[and, implies[p1, p3],
           implies[p2, p4], implies[and[p4, p5], p6], implies[and[p3, p6], p7],
           not[implies[and[p1, p2, p5], p7]],
          {p1 -> subclass[x, y], p2 -> equal[U[x], U[y]],
           p3 -> subclass[composite[complement[inverse[E]], id[x], E],
              composite[complement[inverse[E]], id[y], E]],
           p4 -> equal[composite[Di, id[U[x]]], composite[Di, id[U[y]]]],
           p5 ->
             subclass[composite[Di, id[U[x]]], composite[complement[inverse[E]], id[x], E]],
           p6 ->
             subclass[composite[Di, id[U[y]]], composite[complement[inverse[E]], id[x], E]],
           p7 -> subclass[composite[Di, id[U[y]]],
              composite[complement[inverse[E]], id[y], E]]}]]

Out[49]= or[not[equal[U[x], U[y]]], not[subclass[x, y]], not[
            subclass[composite[Di, id[U[x]]], composite[complement[inverse[E]], id[x], E]]],
           subclass[composite[Di, id[U[y]]], composite[complement[inverse[E]], id[y], E]]] ==
          True
```

This has a simple variable-free formulation:

```
In[50]:= Map[equal[0, #]&, Map[class[pair[x, y], not[#]]&, %]]

Out[50]= subclass[image[COARSER, T1], T1] == True

In[51]:= subclass[image[COARSER, T1], T1] := True
```

■ composites with S and E

This section contains formulas for composites of **COARSER** with **S** and **E**.

```
In[52]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
          {u -> Id, v -> COARSER, w -> S}]

Out[52]= subclass[S, composite[COARSER, S]] == True

In[53]:= subclass[S, composite[COARSER, S]] := True
```

The **GOEDEL** program is able to determine automatically that the reverse inclusion also holds. So we obtain an equation:

```
In[54]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> S, v -> composite[COARSER, S]}]

Out[54]= True == equal[S, composite[COARSER, S]]

In[55]:= composite[COARSER, S] := S
```

The same argument can be repeated with the factors switched:

```
In[56]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
          {u -> Id, v -> COARSER, w -> S}]

Out[56]= subclass[S, composite[S, COARSER]] == True

In[57]:= subclass[S, composite[S, COARSER]] := True

In[58]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> S, v -> composite[S, COARSER]}]

Out[58]= True == equal[S, composite[S, COARSER]]

In[59]:= composite[S, COARSER] := S
```

The assoicative law for **composite** yields a corollary:

```
In[60]:= Assoc[COARSER, S, E]

Out[60]= composite[COARSER, E] == E

In[61]:= composite[COARSER, E] := E
```

The following discovery was made using **RelnRenormality**:

```
In[62]:= composite[inverse[S], COARSER] // RelnRenormality

Out[62]= composite[inverse[S], COARSER] == composite[inverse[S], POWER, BIGCUP]

In[63]:= composite[inverse[S], COARSER] := composite[inverse[S], POWER, BIGCUP]
```

Again, the associative law yields a corollary:

```
In[64]:= Assoc[inverse[E], inverse[S], COARSER] // Reverse

Out[64]= composite[inverse[E], COARSER] == composite[inverse[S], BIGCUP]

In[65]:= composite[inverse[E], COARSER] := composite[inverse[S], BIGCUP]
```

■ some further results

Another application of normality:

```
In[66]:= Map[equal[0, #]&, dif[composite[POWER, BIGCUP], COARSER] // VSNormality]

Out[66]= subclass[composite[POWER, BIGCUP], COARSER] == True

In[67]:= subclass[composite[POWER, BIGCUP], COARSER] := True
```

The final discovery does not involve **COARSER**, but it is of a similar nature.

```
In[68]:= equiv[and[subclass[x, y], subclass[U[x], U[y]]], subclass[x, y]]

Out[68]= True
```

```
In[69]:= and[subclass[x_, y_], subclass[U[x_], U[y_]]] := subclass[x, y]

In[70]:= subclass[inverse[S], composite[inverse[S], POWER, BIGCUP]] // AssertTest

Out[70]= subclass[inverse[S], composite[inverse[S], POWER, BIGCUP]] == True

In[71]:= subclass[inverse[S], composite[inverse[S], POWER, BIGCUP]] := True
```