COMPACT

Johan G. F. Belinfante
2004 February 7

```
In[1]:= << goedel54.05c; << tools.m

        :Package Title: goedel54.05c      2004 February 5 at 9:35 p.m.

        It is now:  2004 Feb 9 at 13:27

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 January 3

        weightlimit = 40
```

summary

For simplicity, the concept of a compact topology will be generalized by omitting the hypothesis that the collection of sets satisfy the axioms for a topology. A collection **t** belongs to the class **COMPACT** if every collection coarser than **t** contains an even coarser finite collection. The **GOEDEL** program currently contains nothing beyond the following wrapped membership rule for this class.

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= InfoMatch[class[w_, HoldPattern[member[x_, COMPACT]]]] // First

Out[3]= {class[w_, member[t_, COMPACT]] := Module[{x = Unique[], y = Unique[]}, class[w, and[
           member[t, V], forall[x, implies[and[subclass[x, t], subclass[U[t], U[x]]],
           exists[y, and[member[y, FINITE], subclass[y, x], subclass[U[x], U[y]]]]]]]]]}
```

Among the basic properties of this class **COMPACT** derived in this notebeook are that finite collections are compact, and that if a discrete topology **P[x]** is compact, then the space **x** is finite.

Normalization

One rule is needed for normalization:

```
In[4]:= complement[COMPACT] // Normality // Reverse

Out[4]= image[COARSER, complement[image[COARSER, FINITE]]] == complement[COMPACT]

In[5]:= image[COARSER, complement[image[COARSER, FINITE]]] := complement[COMPACT]
```

COARSER properties

In this section two basic properties of the relation **COARSER** are derived. This is the first:

```
In[6]:= SubstTest[implies, subclass[u, v],
         subclass[image[u, x], image[v, x]], {u -> Id, v -> inverse[COARSER]}]

Out[6]= subclass[x, image[inverse[COARSER], x]] == True

In[7]:= subclass[x_, image[inverse[COARSER], x_]] := True
```

The second fact that will be needed shortly is:

```
In[8]:= ImageComp[COARSER, COARSER, x] // Reverse

Out[8]= image[COARSER, image[COARSER, x]] == image[COARSER, x]

In[9]:= image[COARSER, image[COARSER, x_]] := image[COARSER, x]
```

the class image[COARSER, FINITE]

The class of all collections of sets which are refinements of finite collections is:

```
In[10]:= class[x, exists[y, and[member[y, FINITE], subclass[y, x], subclass[U[x], U[y]]]]]

Out[10]= image[COARSER, FINITE]
```

It will be shown that the class **COMPACT** is intermediate between **FINITE** and this class **image[COARSER,FINITE]**. The easy part is this:

```
In[11]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
             {u -> Id, v -> COARSER, w -> complement[image[COARSER, FINITE]]}]

Out[11]= subclass[COMPACT, image[COARSER, FINITE]] == True

In[12]:= subclass[COMPACT, image[COARSER, FINITE]] := True
```

COARSER and FINITE

Since any subset of a finite set is finite, it follows that any collection coarser than a finite collection must also be finite.

```
In[13]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
             {u -> COARSER, v -> S, w -> complement[FINITE]}]

Out[13]= subclass[image[inverse[COARSER], FINITE], FINITE] == True

In[14]:= % /. Equal -> SetDelayed
```

This can be strengthened to an equation:

```
In[15]:= equal[image[inverse[COARSER], FINITE], FINITE] // AssertTest

Out[15]= equal[FINITE, image[inverse[COARSER], FINITE]] == True

In[16]:= image[inverse[COARSER], FINITE] := FINITE
```

finite collections are compact

In this section it is shown that the class **FINITE** is a subclass of **COMPACT**. The main difficulty here is contending with rewrite rules that eliminate **complement**. The following temporary rewrite rule will immediately be replaced by a better one:

```
In[17]:= SubstTest[subclass, y, image[COARSER, y], y -> complement[FINITE]]

Out[17]= equal[V, union[FINITE, image[COARSER, complement[FINITE]]]] == True

In[18]:= union[FINITE, image[COARSER, complement[FINITE]]] := V
```

This is the better rule:

```
In[19]:= equal[image[COARSER, complement[FINITE]], complement[FINITE]] // AssertTest

Out[19]= equal[complement[FINITE], image[COARSER, complement[FINITE]]] == True

In[20]:= image[COARSER, complement[FINITE]] := complement[FINITE]
```

It follows that finite collections are compact.

```
In[21]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
             {u -> complement[image[COARSER, FINITE]], v -> complement[FINITE], w -> COARSER}]

Out[21]= subclass[FINITE, COMPACT] == True

In[22]:= subclass[FINITE, COMPACT] := True
```

any collection coarser than a compact one is compact

The normalization rule and the idempotence of the relation **COARSER** imply:

```
In[23]:= ImageComp[COARSER, COARSER, complement[image[COARSER, FINITE]]] // Reverse

Out[23]= image[COARSER, complement[COMPACT]] == complement[COMPACT]

In[24]:= image[COARSER, complement[COMPACT]] := complement[COMPACT]
```

It follows that any collection coarser than a compact collection must also be compact:

```
In[25]:= (equal[image[COARSER, complement[x]], complement[x]] // AssertTest // Reverse) /.
          x -> COMPACT

Out[25]= subclass[image[inverse[COARSER], COMPACT], COMPACT] == True

In[26]:= % /. Equal -> SetDelayed
```

This can be strengthened to an equation:

```
In[27]:= equal[image[inverse[COARSER], COMPACT], COMPACT] // AssertTest

Out[27]= equal[COMPACT, image[inverse[COARSER], COMPACT]] == True

In[28]:= image[inverse[COARSER], COMPACT] := COMPACT
```

compact collections of finite sets

In this section it is shown that a compact collection of finite sets is a finite collection. The following lemma is needed:

```
In[29]:= Map[or[member[t, FINITE], #] &, SubstTest[implies,
           and[equal[x, y], member[y, z]], member[x, z], {x -> U[t], y -> U[s], z -> FINITE}]]

Out[29]= or[member[t, FINITE], not[equal[U[s], U[t]]],
           not[member[s, FINITE]], not[subclass[s, FINITE]]] == True

In[30]:= (% /. {s -> s_, t -> t_}) /. Equal -> SetDelayed
```

If **t** is a collection of finite sets, and if there is a finite collection **s** coarser than **t**, then **t** must be a finite collection:

```
In[31]:= Map[not, SubstTest[and, implies[and[p1, p2], p5],
           implies[and[p3, p4, p5], p6], not[implies[and[p1, p2, p3, p4], p6]],
           {p1 -> subclass[t, FINITE], p2 -> subclass[s, t], p3 -> equal[U[s], U[t]],
            p4 -> member[s, FINITE], p5 -> subclass[s, FINITE], p6 -> member[t, FINITE]}]]

Out[31]= or[member[t, FINITE], not[equal[U[s], U[t]]],
           not[member[s, FINITE]], not[subclass[s, t]], not[subclass[t, FINITE]]] == True

In[32]:= (% /. {s -> s_, t -> t_}) /. Equal -> SetDelayed
```

Removing the variables **s** and **t** yields a variable−free reformulation of this fact:

```
In[33]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class, pair[s, t],
            implies[and[member[pair[s, t], x], member[s, y], subclass[t, y]], member[t, y]],
             {x -> COARSER, y -> FINITE}]] // Reverse

Out[33]= subclass[intersection[image[COARSER, FINITE], P[FINITE]], FINITE] == True

In[34]:= % /. Equal -> SetDelayed
```

This is equivalent:

```
In[35]:= equal[intersection[image[COARSER, FINITE], P[FINITE]],
           intersection[FINITE, P[FINITE]]] // AssertTest

Out[35]= equal[intersection[FINITE, P[FINITE]],
           intersection[image[COARSER, FINITE], P[FINITE]]] == True

In[36]:= intersection[image[COARSER, FINITE], P[FINITE]] := intersection[FINITE, P[FINITE]]
```

Corollary: a compact collection of finite sets is finite.

```
In[37]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
             {u -> COMPACT, v -> image[COARSER, FINITE], w -> id[dif[P[FINITE], FINITE]]}]

Out[37]= subclass[intersection[COMPACT, P[FINITE]], FINITE] == True

In[38]:= % /. Equal -> SetDelayed
```

This can be reformulated as an equation:

```
In[39]:= equal[intersection[COMPACT, P[FINITE]], intersection[FINITE, P[FINITE]]] // AssertTest

Out[39]= equal[intersection[COMPACT, P[FINITE]], intersection[FINITE, P[FINITE]]] == True

In[40]:= intersection[COMPACT, P[FINITE]] := intersection[FINITE, P[FINITE]]
```

a characterization of compactness

From the wrapped membership rule used to define the class **COMPACT** in the **GOEDEL** program, one can use **AssertTest** to derive an unwrapped membership rule, inspired by the following description of this class:

```
In[41]:= class[x, subclass[image[inverse[COARSER], singleton[x]], image[COARSER, FINITE]]]

Out[41]= COMPACT
```

The following observation suggests a suitable class to which one can apply **AssertTest**.

```
In[42]:= class[x, subclass[image[inverse[y], singleton[x]], z]]

Out[42]= complement[image[y, complement[z]]]
```

The idea is to apply **AssertTest** to the statetment that **x** is a member of this class, and then to make some substitutions:

```
In[43]:= (not[member[x, image[y, complement[z]]]] // AssertTest // Reverse) /.
          {y -> COARSER, z -> image[COARSER, FINITE]}

Out[43]= subclass[image[inverse[COARSER], singleton[x]], image[COARSER, FINITE]] ==
          or[member[x, COMPACT], not[member[x, V]]]
```

The orientation of this rewrite rule has been chosen to prevent it from being expanded each time one wishes to assert that a collection is compact.

```
In[44]:= subclass[image[inverse[COARSER], singleton[x_]], image[COARSER, FINITE]] :=
          or[member[x, COMPACT], not[member[x, V]]]
```

This says that a collection **x** is compact if for every coarser collection, there is an even coarser finite collection. It will be convenient in what follows to introduce an extra variable in this statement, yielding the following corollary:

```
In[45]:= SubstTest[implies, and[member[x, u], subclass[u, v]], member[x, v],
              {u -> image[inverse[COARSER], singleton[y]], v -> image[COARSER, FINITE]}]

Out[45]= or[member[x, image[COARSER, FINITE]], not[equal[U[x], U[y]]],
            not[member[y, COMPACT]], not[subclass[x, y]]] == True
```

This says that if a collection **x** is compact, and if **y** is coarser than **x**, then there is a finite set even coarser than **y**. The converse also holds, but will not be needed.

```
In[46]:= or[member[x_, image[COARSER, FINITE]], not[equal[U[x_], U[y_]]],
            not[member[y_, COMPACT]], not[subclass[x_, y_]]] := True
```

compact discrete spaces

In the case that the compact collection of interest is a power set **P[x]**, the collection **image[SINGLETON,x]** of all singletons of members of **x** is a coarser collection, so one obtains the following result:

```
In[47]:= SubstTest[implies, and[subclass[u, v], equal[U[u], U[v]], member[v, COMPACT]],
          member[u, image[COARSER, FINITE]],
             {u -> image[SINGLETON, x], v -> P[x]}]

Out[47]= or[member[image[SINGLETON, x], image[COARSER, FINITE]],
            not[member[P[x], COMPACT]]] == True

In[48]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The following further consequence of the conclusion is needed:

```
In[49]:= implies[member[image[SINGLETON, x], image[COARSER, FINITE]], member[x, FINITE]] //
          AssertTest

Out[49]= or[member[x, FINITE], not[member[image[SINGLETON, x], image[COARSER, FINITE]]]] == True

In[50]:= (% /. x -> x_) /. Equal -> SetDelayed
```

From this it is easy to derive the fact that the class **FINITE** is also an upper bound:

```
In[51]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p2, p3], not[implies[p1, p3]], {p1 -> member[P[x], COMPACT],
             p2 -> member[image[SINGLETON, x], image[COARSER, FINITE]], p3 -> member[x, FINITE]}]]

Out[51]= or[member[x, FINITE], not[member[P[x], COMPACT]]] == True

In[52]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Eliminating the variable yields:

```
In[53]:= Map[equal[V, #] &, complement[dif[image[inverse[POWER], COMPACT], FINITE]] // Normality]

Out[53]= subclass[image[inverse[POWER], COMPACT], FINITE] == True

In[54]:= % /. Equal -> SetDelayed
```

This can be strengthened to an equation:

```
In[55]:= equal[image[inverse[POWER], COMPACT], FINITE] // AssertTest

Out[55]= equal[FINITE, image[inverse[POWER], COMPACT]] == True

In[56]:= image[inverse[POWER], COMPACT] := FINITE
```

Corollary:

```
In[57]:= SubstTest[member, x, image[inverse[POWER], y], y -> COMPACT] // Reverse

Out[57]= member[P[x], COMPACT] == member[x, FINITE]
```

The collection of all subsets of **x** is a compact topology if and only if **x** is finite.

```
In[58]:= member[P[x_], COMPACT] := member[x, FINITE]
```