Uclosure and compactness

Johan G. F. Belinfante
2011 February 28

```
In[1]:= SetDirectory["l:"]; << goedel.11feb26a

        :Package Title: goedel.11feb26a                 2011 February 26 at 9:20 a.m.

        It is now:  2011 Feb 28 at 15:43

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

In the **GOEDEL** program, the concept of compactness is defined in terms of the relation **COARSER** rather than via coverings. A collection of sets **x** is **coarser** than another collection **y** if **x ⊂ y** and **U[x] = U[y]**.

```
In[2]:= member[pair[x, y], COARSER]

Out[2]= and[equal[U[x], U[y]], member[y, V], subclass[x, y]]
```

A collection of sets is said to be **compact** if for every coarser collection, there is a finite collection that is even coarser.

```
In[3]:= assert[forall[y, implies[member[pair[y, x], COARSER],
           exists[z, and[member[z, FINITE], member[pair[z, y], COARSER]]]]]]

Out[3]= or[member[x, COMPACT], not[member[x, V]]]
```

The concept of a compact collection of sets is not limited to topology. The reader is cautioned that when dealing with topological spaces in the **GOEDEL** program, the term **compact** refers to the collection **t** of open sets rather than to the underlying space **U[t]**. Thus what is ordinarily called a topological space in the literature is here described as a space whose collection of open sets is a compact collection. Even in topology, the concept of a compact collection of sets is not limited to collections of open sets, and for instance could also be applied to an open basis or a subbasis.

The **Uclosure** of a collection of sets is the collection of sets obtained by forming unions of arbitrary subcollections.

```
In[4]:= class[u, exists[s, and[subclass[s, x], equal[u, U[s]]]]]

Out[4]= Uclosure[x]
```

The main result proved in this notebook is that a collection of sets is compact if and only if its Uclosure is compact. The easy half of this statement was already shown in the **GOEDEL** program in October 2004. It is easy to see that any collec-

tion of sets coarser than a compact collection is also compact. Since any collection of sets is coarser than its Uclosure, it follows immediately that if **Uclosure[x]** is compact, then so is **x** itself.

```
In[5]:= implies[member[Uclosure[x], COMPACT], member[x, COMPACT]]
```

```
Out[5]= True
```

The axiom of choice is not needed for the reverse implication since the only choices that need to be made are finite choices. A brief sketch of the proof will be given before getting into the details. Suppose **x ∈ COMPACT**, and let **y ⊂ Uclosure[x]** be a coarser collection. Then **z = x ⋂ image[inverse[S], y]** is coarser than **x** because each element **t ∈ y ⊂ Uclosure[x]** is a union of some subset of **z**. Since **x** is compact, there is a finite subset **f ⊂ z** that also covers **U[x]**. Since **f** is finite, there is a choice function **g** that selects one element **v ∈ y** for each **u ∈ f** such that **u ⊂ v**. The selected elements **v** form a finite collection coarser than **y**. Hence **Uclosure[x] ∈ COMPACT**.

---

application of finite choice

The theorem of finite choice says that any relation with a finite domain has a cross-section. The **fin** wrapper will be used for the finite set **f**.

Theorem. (Application of the theorem of finite choice.)

```
In[6]:= Map[not, SubstTest[implies, member[domain[t], FINITE],
          not[empty[X[t]]], t → composite[id[y], S, id[fin[f]]]]] // Reverse
```

```
Out[6]= equal[0, X[composite[id[y], S, id[fin[f]]]]] == False
```

```
In[7]:= equal[0, X[composite[id[y_], S, id[fin[f_]]]]] := False
```

An auxiliary variable **g** will be introduced for the cross-section of the relation **id[y] ∘ S ∘ id[fin[f]]**.

Lemma.

```
In[8]:= or[member[t, map[x, y]], not[member[t, map[x, V]]],
         not[subclass[t, cart[x, y]]]] // NotNotTest
```

```
Out[8]= or[member[t, map[x, y]], not[member[t, map[x, V]]], not[subclass[t, cart[x, y]]]] == True
```

```
In[9]:= or[member[t_, map[x_, y_]],
         not[member[t_, map[x_, V]]], not[subclass[t_, cart[x_, y_]]]] := True
```

Lemma.

```
In[10]:= SubstTest[implies, and[equal[u, v], member[g, map[u, V]], subclass[g, cart[v, y]]],
          member[g, map[v, y]], u → intersection[v, w]] // Reverse
```

```
Out[10]= or[member[g, map[v, y]], not[member[g, map[intersection[v, w], V]]],
           not[subclass[g, cart[v, y]]], not[subclass[v, w]]] == True
```

```
In[11]:= or[member[g_, map[v_, y_]], not[member[g_, map[intersection[v_, w_], V]]],
           not[subclass[g_, cart[v_, y_]]], not[subclass[v_, w_]]] := True
```

Restatement:

```
In[12]:= implies[and[subclass[fin[f], image[inverse[S], y]],
           member[g, X[composite[id[y], S, id[fin[f]]]]]], member[g, map[fin[f], y]]]

Out[12]= True
```

Lemma.

```
In[13]:= SubstTest[implies, subclass[x, t],
           subclass[U[x], U[t]], t → image[inverse[S], y]] // Reverse

Out[13]= or[not[subclass[x, image[inverse[S], y]]], subclass[U[x], U[y]]] == True

In[14]:= or[not[subclass[x_, image[inverse[S], y_]]], subclass[U[x_], U[y_]]] := True
```

Lemma.

```
In[15]:= SubstTest[implies, subclass[g, w], subclass[image[t, g], image[t, w]],
           {t → composite[inverse[E], SECOND], w -> composite[id[y], S, id[fin[f]]]}] // Reverse

Out[15]= or[not[subclass[g, S]], not[subclass[g, cart[fin[f], y]]],
           subclass[U[range[g]], U[intersection[y, image[S, fin[f]]]]] == True

In[16]:= (% /. {g → g_, f → f_, y → y_}) /. Equal→ SetDelayed
```

Theorem.

```
In[17]:= Map[not, SubstTest[and, implies[p1, p4],
            implies[p2, p3], implies[and[p0, p4], p5], implies[and[p3, p5], p6],
            not[implies[and[p0, p1, p2], p6]], {p0 → subclass[fin[f], image[inverse[S], y]],
             p1 -> member[g, map[intersection[fin[f], image[inverse[S], y]], V]],
             p2 -> subclass[g, S], p3 -> subclass[U[domain[g]], U[range[g]]],
             p4 → equal[domain[g], intersection[fin[f], image[inverse[S], y]]],
             p5 → equal[domain[g], fin[f]], p6 -> subclass[U[fin[f]], U[range[g]]]}]] // Reverse

Out[17]= or[not[member[g, map[intersection[fin[f], image[inverse[S], y]], V]]],
           not[subclass[g, S]], not[subclass[fin[f], image[inverse[S], y]]],
           subclass[U[fin[f]], U[range[g]]]] == True

In[18]:= (% /. {g → g_, f → f_, y → y_}) /. Equal→ SetDelayed
```

Corollary. **U[range[g]] ⊂ U[y]**.

```
In[19]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[p3, p4], not[implies[and[p1, p2], p4]],
            {p1 -> subclass[g, S], p2 -> subclass[g, cart[fin[f], y]],
             p3 -> subclass[U[range[g]], U[intersection[y, image[S, fin[f]]]]],
             p4 -> subclass[U[range[g]], U[y]]}]] // Reverse

Out[19]= or[not[subclass[g, S]],
           not[subclass[g, cart[fin[f], y]]], subclass[U[range[g]], U[y]]] == True
```

```
In[20]:= or[not[subclass[g_, S]],
            not[subclass[g_, cart[fin[f_], y_]]], subclass[U[range[g_]], U[y_]]] := True
```

Lemma.

```
In[21]:= SubstTest[implies, and[equal[u, v], member[u, V]],
            member[v, V], {u → U[fin[f]], v → U[y]}] // Reverse
```

```
Out[21]= or[member[y, V], not[equal[U[y], U[fin[f]]]]] == True
```

```
In[22]:= (% /. {f → f_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[23]:= Map[not, SubstTest[and, implies[and[p0, p1], p3], implies[p3, p4],
             not[implies[and[p0, p1], p4]], {p0 -> member[g, X[composite[id[y], S, id[fin[f]]]]],
              p1 → subclass[fin[f], image[inverse[S], y]], p2 → equal[U[y], U[fin[f]]],
              p3 -> member[g, map[fin[f], y]], p4 -> member[range[g], FINITE]}]] // Reverse
```

```
Out[23]= or[member[range[g], FINITE],
            not[member[g, map[intersection[fin[f], image[inverse[S], y]], V]]],
            not[subclass[g, S]], not[subclass[g, cart[fin[f], y]]],
            not[subclass[fin[f], image[inverse[S], y]]]] == True
```

```
In[24]:= (% /. {f → f_, g → g_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[26]:= Map[not, SubstTest[and, implies[and[p1, p3], p4], implies[and[p1, p2, p3, p4, p5], p6],
             not[implies[and[p1, p2, p3, p5], p6]], {p1 → equal[U[y], U[fin[f]]],
              p2 -> member[range[g], FINITE], p3 -> subclass[range[g], y],
              p4 → subclass[U[range[g]], U[fin[f]]], p5 → subclass[U[fin[f]], U[range[g]]],
              p6 → member[y, image[COARSER, FINITE]]}]] // Reverse
```

```
Out[26]= or[member[y, image[COARSER, FINITE]],
            not[equal[U[y], U[fin[f]]]], not[member[range[g], FINITE]],
            not[subclass[range[g], y]], not[subclass[U[fin[f]], U[range[g]]]]] == True
```

```
In[27]:= (% /. {f → f_, g → g_, y → y_}) /. Equal → SetDelayed
```

Omitted proof steps in the following derivation are indicated using *Mathematica*'s comment notation **(* ... *)**.

Theorem.

```
In[28]:= Map[not, SubstTest[and,
           (*implies[and[p0,p1],p3], implies[and[p0,p1],p4],implies[and[p0,p1],p5], *)
           implies[and[p2, p3, p4, p5], p6], not[implies[and[p0, p1, p2], p6]] ,
           {p0 -> member[g, X[composite[id[y], S, id[fin[f]]]]],
            p1 → subclass[fin[f], image[inverse[S], y]],
            p2 → equal[U[y], U[fin[f]]], p3 -> member[range[g], FINITE],
            p4 -> subclass[range[g], y], p5 → subclass[U[fin[f]], U[range[g]]],
            p6 → member[y, image[COARSER, FINITE]]}]] // Reverse

Out[28]= or[member[y, image[COARSER, FINITE]], not[equal[U[y], U[fin[f]]]],
           not[member[g, map[intersection[fin[f], image[inverse[S], y]], V]]],
           not[subclass[g, S]], not[subclass[g, cart[fin[f], y]]],
           not[subclass[fin[f], image[inverse[S], y]]]] == True

In[29]:= (% /. {f → f_, g → g_, y → y_}) /. Equal → SetDelayed
```

Theorem. (Eliminating the cross-section variable **g**.)

```
In[30]:= Map[equal[V, #] &, SubstTest[class, g,
           implies[and[member[g, x], subclass[t, image[inverse[S], y]], equal[U[y], U[t]]],
            member[y, image[COARSER, FINITE]],
           {t → fin[f], x -> X[composite[id[y], S, id[fin[f]]]]}]]

Out[30]= or[member[y, image[COARSER, FINITE]], not[equal[U[y], U[fin[f]]]],
           not[subclass[fin[f], image[inverse[S], y]]]] == True

In[31]:= (% /. {f → f_, y → y_}) /. Equal → SetDelayed
```

Lemma.. Eliminating the **fin** wrapper.

```
In[32]:= SubstTest[implies, equal[f, fin[t]], or[member[y, image[COARSER, FINITE]],
            not[equal[U[y], U[f]]], not[subclass[f, image[inverse[S], y]]]], t → f] // Reverse

Out[32]= or[member[y, image[COARSER, FINITE]], not[equal[U[f], U[y]]],
           not[member[f, FINITE]], not[subclass[f, image[inverse[S], y]]]] == True

In[33]:= (% /. {f → f_, y → y_}) /. Equal → SetDelayed
```

---

the rest of the proof

Theorem. If **y** is coarser than **Uclosure[x]**, then **z = x ⋂ image[inverse[S], y]** is coarser than **x**.

```
In[34]:= SubstTest[implies, equal[y, image[funpart[t], y]],
           equal[U[y], U[image[funpart[t], y]]], t → CORE[x]] // Reverse

Out[34]= or[equal[U[y], U[intersection[x, image[inverse[S], y]]]],
           not[subclass[y, Uclosure[x]]]] == True

In[35]:= or[equal[U[y_], U[intersection[x_, image[inverse[S], y_]]]],
           not[subclass[y_, Uclosure[x_]]]] := True
```

Lemma. If **x** is compact, and **z** is coarser than **x**, then there is a finite set coarser than **z**.

```
In[36]:= SubstTest[implies, and[member[z, u], subclass[u, v]], member[z, v],
           {u → image[inverse[COARSER], set[x]], v → image[COARSER, FINITE],
            z -> intersection[x, image[inverse[S], y]]}] // Reverse

Out[36]= or[member[intersection[x, image[inverse[S], y]], image[COARSER, FINITE]],
           not[equal[U[x], U[intersection[x, image[inverse[S], y]]]]],
           not[member[x, COMPACT]]] == True

In[37]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. If **x** is compact and **y** is coarser than **Uclosure[x]** then there is a finite set coarser than **z = x ⋂ image[inverse[S], y]**.

```
In[38]:= Map[not, SubstTest[and, implies[p2, p4], implies[and[p3, p4], p5],
             implies[and[p1, p5], p6], not[implies[and[p1, p2, p3], p6]],
             {p1 → member[x, COMPACT], p2 → subclass[y, Uclosure[x]], p3 → equal[U[x], U[y]],
              p4 -> equal[U[y], U[intersection[x, image[inverse[S], y]]]],
              p5 -> equal[U[x], U[intersection[x, image[inverse[S], y]]]], p6 -> member[
                intersection[x, image[inverse[S], y]], image[COARSER, FINITE]]}]] // Reverse

Out[38]= or[member[intersection[x, image[inverse[S], y]], image[COARSER, FINITE]],
           not[equal[U[x], U[y]]], not[member[x, COMPACT]], not[subclass[y, Uclosure[x]]]] == True

In[39]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[40]:= Map[not, SubstTest[and, implies[and[p2, p3, p4], p9],
             implies[and[p5, p8, p9], p10], not[implies[and[p2, p3, p4, p5, p8], p10]],
             {p2 → equal[U[x], U[y]], p3 → subclass[y, Uclosure[x]],
              p4 -> equal[U[f], U[intersection[x, image[inverse[S], y]]]],
              p5 -> member[f, FINITE], p8 -> subclass[f, image[inverse[S], y]],
              p9 → equal[U[f], U[y]], p10 -> member[y, image[COARSER, FINITE]]}]] // Reverse

Out[40]= or[member[y, image[COARSER, FINITE]],
           not[equal[U[f], U[intersection[x, image[inverse[S], y]]]]],
           not[equal[U[x], U[y]]], not[member[f, FINITE]],
           not[subclass[f, image[inverse[S], y]]], not[subclass[y, Uclosure[x]]]] == True

In[41]:= (% /. {x → x_, y → y_, f → f_}) /. Equal → SetDelayed
```

Restatement.

```
In[42]:= implies[and[subclass[y, Uclosure[x]], equal[U[x], U[y]], member[f, FINITE],
           member[pair[f, intersection[x, image[inverse[S], y]]], COARSER]],
          member[y, image[COARSER, FINITE]]]

Out[42]= True
```

Theorem. Eliminate the variable **f**.

```
In[43]:= Map[equal[V, #] &,
          SubstTest[class, f, implies[and[subclass[y, u], equal[U[x], U[y]], member[f, FINITE],
            member[pair[f, intersection[x, image[inverse[S], y]]], t]], member[y, z]],
           {t → COARSER, u→ Uclosure[x], z → image[COARSER, FINITE]}]]

Out[43]= or[member[y, image[COARSER, FINITE]], not[equal[U[x], U[y]]],
           not[member[intersection[x, image[inverse[S], y]], image[COARSER, FINITE]]],
           not[subclass[y, Uclosure[x]]]] == True

In[44]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

Theorem. If **x** is compact and **y** is coarser than **Uclosure[x]**, then there is a finite set coarser than **y**.

```
In[45]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 → member[x, COMPACT], p2 -> member[pair[y, Uclosure[x]], COARSER],
             p3 -> member[intersection[x, image[inverse[S], y]], image[COARSER, FINITE]],
             p4 -> member[y, image[COARSER, FINITE]]}]] // Reverse

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

In[46]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

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

```
In[47]:= Map[equal[V, #] &, SubstTest[class, y,
           implies[and[member[x, COMPACT], member[pair[y, Uclosure[x]], t]], member[y, v]],
           {t → COARSER, u→ COMPACT, v→ image[COARSER, FINITE]}]]

Out[47]= or[member[Uclosure[x], COMPACT], not[member[x, COMPACT]]] == True

In[48]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

A better rewrite rule is obtained by combining the above result with the converse statement.

Corollary.

```
In[49]:= equiv[member[Uclosure[x], COMPACT], member[x, COMPACT]]

Out[49]= True

In[50]:= member[Uclosure[x_], COMPACT] := member[x, COMPACT]
```

Theorem. A variable-free restatement of the theorem.

```
In[51]:= image[inverse[UCLOSURE], COMPACT] // Normality

Out[51]= image[inverse[UCLOSURE], COMPACT] == COMPACT

In[52]:= image[inverse[UCLOSURE], COMPACT] := COMPACT
```

Corollary.

In[53]:= **ImageComp[UCLOSURE, inverse[UCLOSURE], COMPACT] // Reverse**

Out[53]= image[UCLOSURE, COMPACT] == intersection[COMPACT, fix[UCLOSURE]]

In[54]:= **image[UCLOSURE, COMPACT] := intersection[COMPACT, fix[UCLOSURE]]**