PO ⊂ image[inverse[S], TO]

Johan G. F. Belinfante
2007 December 30

In[1]:= **SetDirectory["l:"]; << goedel.07dec29a; << tools.m**

```
:Package Title: goedel.07dec29a          2007 December 29 at 7:00 p.m.

It is now:  2007 Dec 30 at 3:52

Loading Simplification Rules

TOOLS.M                                   Revised 2007 December 29

weightlimit = 40
```

introduction

One application of Zorn's lemma is the Szpilrajn theorem, which says that any partial order can be extended to a total order. For this application, the class to which Zorn's lemma is applied is not the class **PO** of all partial orders, but rather the class of all partial orders contained in a fixed cartesian square.

In[2]:= **"E. Szpilrajn, *On the Extension of a Partial Ordering*, Fund. Math., vol. 16, pp. 386-389 (1930). [in French]";**

To prove this theorem, one needs to show that a maximal partial order on a given set is a total order. If **x** is a partial order and if an ordered pair **⟨u,v⟩** does not belong to **x**, then one can extend the partial order **x** one by adjoining to **x** the cartesian product of **image[inverse[x], set[u]]** and **image[x, set[v]]**. This extension is proper if the ordered pair **⟨v,u⟩** also does not belong to **x**.

adjoining an identity

A partial order cannot be maximal if it can be extended by adjoining an identity. In this section it is shown that if a partial order **x** is maximal in the cartesian square **cart[y, y]**, then **y = fix[x]**.

Lemma. We begin by wrapping **x** with **po**. Note that the desired equation **y = fix[po[x]]** is here replaced by an equivalent inclusion. The use of the **po** wrapper causes the inclusion of the partial order in **cart[y, y]** to be rewritten as **subclass[fix[po[x]],y]**.

```
In[3]:= (SubstTest[or, equal[r, s],
          member[r, image[inverse[PS], t]], not[member[s, t]], not[subclass[r, s]],
          {r → po[x], s → union[id[y], po[x]], t -> intersection[PO, P[cart[y, y]]]}] // Reverse)

Out[3]= or[member[po[x], image[inverse[PS], intersection[PO, P[cart[y, y]]]]],
          not[member[y, V]], not[member[po[x], V]],
          not[subclass[fix[po[x]], y]], subclass[y, fix[po[x]]]] == True

In[4]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

One needs to remove the **po** wrapper before proceeding further.

```
In[5]:= SubstTest[implies, equal[x, po[t]],
          or[member[x, image[inverse[PS], intersection[PO, P[cart[y, y]]]]], not[member[y, V]],
           not[member[x, V]], not[subclass[fix[x], y]], subclass[y, fix[x]]], t → x] // Reverse

Out[5]= or[member[x, image[inverse[PS], intersection[PO, P[cart[y, y]]]]],
          not[member[x, V]], not[member[y, V]], not[PARTIALORDER[x]],
          not[subclass[fix[x], y]], subclass[y, fix[x]]] == True

In[6]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The condition **subclass[fix[x],y]** can now be replaced with **subclass[x,cart[y,y]]**. At the same time, the equation **y = fix[x]** can be made explicit.

```
In[7]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p0, p1, p2], p3],
          implies[and[p2, p3], p4], not[implies[and[p0, p1], p4]],
          {p0 → member[y, V], p1 → member[x, maximal[S, intersection[PO, P[cart[y, y]]]]],
           p2 → subclass[fix[x], y], p3 → subclass[y, fix[x]], p4 → equal[y, fix[x]]}]] // Reverse

Out[7]= or[equal[y, fix[x]],
          member[x, image[inverse[PS], intersection[PO, P[cart[y, y]]]]], not[member[x, V]],
          not[member[y, V]], not[PARTIALORDER[x]], not[subclass[x, cart[y, y]]]] == True

In[8]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

---

adjoining a cartesian product

A partial order cannot be maximal if it can be extended by adjoining a cartesian product. In this section it is shown that if a partial order **x** is maximal in the cartesian square **cart[y, y]**, then **x** is a total order. One needs to consider separately the transitive, reflexive and antisymmetric properties for extensions of a partial order by a (certain type of) cartesian product. For convenience only the case **y = fix[x]** is considered for the initial lemmas, thereby reducing the number of free variables.

Lemma. For the transitive property, the following suffices.

```
In[9]:= SubstTest[subclass, composite[t, t], t,
         t -> union[po[x], cart[image[inverse[po[x]], u], image[po[x], v]]]]

Out[9]= TRANSITIVE[union[cart[image[inverse[po[x]], u], image[po[x], v]], po[x]]] == True
```

```
In[10]:= (% /. {x → x_, u → u_, v → v_}) /. Equal → SetDelayed
```

Lemma.

```
In[12]:= SubstTest[REFLEXIVE, union[id[x], rfx[t]], t → po[y]] // Reverse

Out[12]= REFLEXIVE[union[id[x], po[y]]] == True

In[13]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. For the reflexive property, the following suffices.

```
In[14]:= SubstTest[subclass, t, cartsq[fix[t]],
          t -> union[cart[image[inverse[po[x]], y], image[po[x], z]], po[x]]]

Out[14]= REFLEXIVE[union[cart[image[inverse[po[x]], y], image[po[x], z]], po[x]]] == True

In[15]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

For the antisymmetric property, two lemmas are needed. This is the first:

```
In[17]:= SubstTest[implies, empty[t], subclass[t, Id], t -> composite[
             id[image[inverse[po[x]], set[z]]], po[x], id[image[po[x], set[y]]]]] // Reverse

Out[17]= or[member[pair[y, z], po[x]], subclass[composite[
             id[image[inverse[po[x]], set[z]]], po[x], id[image[po[x], set[y]]]], Id]] == True

In[18]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The second lemma is similar, but involves **inverse[po[x]]** instead of **po[x]**.

```
In[19]:= SubstTest[implies, empty[t], subclass[t, Id], t → composite[id[image[po[x], set[y]]],
             inverse[po[x]], id[image[inverse[po[x]], set[z]]]]] // Reverse

Out[19]= or[member[pair[y, z], po[x]], subclass[composite[id[image[po[x], set[y]]],
             inverse[po[x]], id[image[inverse[po[x]], set[z]]]], Id]] == True

In[20]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. Combining the above lemmas, one obtains this result: if **pair[y, z]** does not belong to **po[x]**, then one can extend **po[x]** by adjoining a cartesian product.

```
In[21]:= Map[implies[not[member[pair[y, z], po[x]]], #] &,
           SubstTest[and, REFLEXIVE[t], ANTISYMMETRIC[t], TRANSITIVE[t],
            t -> union[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]]]] //
          MapNotNot

Out[21]= or[member[pair[y, z], po[x]], PARTIALORDER[
            union[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]]]] == True

In[22]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The extension is not proper unless the reverse ordered pair also does not belong to **po[x]**.

```
In[23]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → set[PAIR[z, y]], v -> cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]],
            w -> po[x]}] // Reverse

Out[23]= or[member[pair[z, y], po[x]],
           not[member[y, fix[po[x]]]], not[member[z, fix[po[x]]]], not[
            subclass[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]]]] == True

In[24]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The following lemma introduces the class **image[inverse[PS],intersection[PO,P[cart[y, y]]]]**. Both **po** and **setpart** wrappers are used here.

```
In[25]:= (SubstTest[or, equal[r, s], member[r, image[inverse[PS], t]],
             not[member[s, t]], not[subclass[r, s]], {r → po[x],
              s → union[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]], t ->
               intersection[PO, P[cart[fix[po[x]], fix[po[x]]]]]}] // Reverse) /. x → setpart[t]

Out[25]= or[member[po[setpart[t]], image[inverse[PS],
             intersection[PO, P[cart[fix[po[setpart[t]]], fix[po[setpart[t]]]]]]]],
           not[PARTIALORDER[union[cart[image[inverse[po[setpart[t]]], set[z]],
               image[po[setpart[t]], set[y]]], po[setpart[t]]]]],
           subclass[cart[image[inverse[po[setpart[t]]], set[z]],
             image[po[setpart[t]], set[y]]], po[setpart[t]]]] == True

In[26]:= (% /. {t → t_, y → y_, z → z_}) /. Equal → SetDelayed
```

Removing the **setpart** wrapper yields:

```
In[34]:= SubstTest[implies, equal[x, setpart[t]], or[member[po[x], image[inverse[PS],
              intersection[PO, P[cart[fix[po[x]], fix[po[x]]]]]]], not[PARTIALORDER[
              union[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]]]],
            subclass[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]]], t →
            x] // Reverse

Out[34]= or[member[po[x],
            image[inverse[PS], intersection[PO, P[cart[fix[po[x]], fix[po[x]]]]]]],
           not[member[x, V]], not[PARTIALORDER[
             union[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]]]],
           subclass[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]]] == True

In[35]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

If neither **pair[y,z]** nor **pair[z,y]** belongs to **po[x]**, and **y = fix[po[x]]**, then **po[x]** is not a maximal element of **intersection[PO, P[cart[y, y]]]**.

```
In[36]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p1, p2, p3], p4], not[implies[p1, p4]],
            {p1 → and[member[x, V], member[y, fix[po[x]]], member[z, fix[po[x]]], not[
                member[pair[y, z], po[x]]], not[member[pair[z, y], po[x]]]], p2 -> PARTIALORDER[
              union[cart[image[inverse[po[x]], set[z]], image[po[x], set[y]]], po[x]]],
             p3 → not[subclass[cart[image[inverse[po[x]], set[z]],
                 image[po[x], set[y]]], po[x]]],
             p4 -> member[po[x], image[inverse[PS], intersection[PO,
                 P[cart[fix[po[x]], fix[po[x]]]]]]]}]] // Reverse

Out[36]= or[member[pair[y, z], po[x]], member[pair[z, y], po[x]], member[po[x],
            image[inverse[PS], intersection[PO, P[cart[fix[po[x]], fix[po[x]]]]]]],
          not[member[x, V]], not[member[y, fix[po[x]]]], not[member[z, fix[po[x]]]]] == True

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

Next the variables **y** and **z** are eliminated. For convenience, the **setpart** wrapper is reintroduced here.

```
In[43]:= (Map[empty[composite[Id, complement[#]]] &,
            SubstTest[class, pair[y, z], or[member[pair[y, z], t], member[pair[z, y], t],
              member[t, u], not[member[x, V]], not[member[y, v]], not[member[z, v]]], {t → po[x],
              u -> image[inverse[PS], intersection[PO, P[cart[fix[po[x]], fix[po[x]]]]]],
              v → fix[po[x]]}]]) /. x → setpart[t]

Out[43]= or[member[po[setpart[t]], image[inverse[PS],
             intersection[PO, P[cart[fix[po[setpart[t]]], fix[po[setpart[t]]]]]]]],
          TOTALORDER[po[setpart[t]]]] == True

In[44]:= (% /. t → t_) /. Equal -> SetDelayed
```

Removing both the **po** and **setpart** wrappers yields:

```
In[45]:= SubstTest[implies, equal[x, po[setpart[t]]],
          or[member[x, image[inverse[PS], intersection[PO, P[cart[fix[x], fix[x]]]]]],
            TOTALORDER[x]], t → x] // Reverse

Out[45]= or[member[x, image[inverse[PS], intersection[PO, P[cart[fix[x], fix[x]]]]]],
          not[member[x, V]], not[PARTIALORDER[x]], TOTALORDER[x]] == True

In[46]:= (% /. x → x_) /. Equal → SetDelayed
```

This result is now combined with that of the preceding section.

```
In[47]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
            {p1 → and[member[x, maximal[S, intersection[PO, P[cart[y, y]]]]], member[y, V]],
             p2 → equal[y, fix[x]], p3 → TOTALORDER[x]}]] // Reverse

Out[47]= or[member[x, image[inverse[PS], intersection[PO, P[cart[y, y]]]]],
          not[member[x, V]], not[member[y, V]], not[PARTIALORDER[x]],
          not[subclass[x, cart[y, y]]], TOTALORDER[x]] == True

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

Finally, we eliminate the variable **x**.

```
In[49]:= Map[equal[V, #] &, SubstTest[class, x, or[member[x, u], not[member[x, v]],
            not[member[y, V]], not[subclass[x, cart[y, y]]], member[x, w]],
           {u -> image[inverse[PS], intersection[PO, P[cart[y, y]]]], v → PO, w → TO}]]

Out[49]= or[not[member[y, V]], subclass[intersection[PO, P[cart[y, y]]],
            union[TO, image[inverse[PS], intersection[PO, P[cart[y, y]]]]]]] == True

In[50]:= (% /. y → y_) /. Equal → SetDelayed
```

applying Zorn's lemma

A strong version of Zorn's lemma is applied to the intersection of **PO** with the power set of a cartesian square, using **setpart** wrappers.

```
In[51]:= SubstTest[implies, and[axch, member[t, V], subclass[Uchains[t], image[inverse[S], t]]],
           subclass[t, image[inverse[S], intersection[t, complement[image[inverse[PS], t]]]]],
           t -> intersection[PO, P[cartsq[setpart[y]]]]] // Reverse

Out[51]= or[not[axch], subclass[intersection[PO, P[cart[setpart[y], setpart[y]]]],
            image[inverse[S], intersection[PO, complement[
               image[inverse[PS], intersection[PO, P[cart[setpart[y], setpart[y]]]]]],
             P[cart[setpart[y], setpart[y]]]]]]]] == True

In[52]:= (% /. y → y_) /. Equal → SetDelayed
```

The **setpart** wrapper is now removed:

```
In[53]:= SubstTest[implies, equal[x, setpart[y]],
           or[not[axch], subclass[intersection[PO, P[cart[x, x]]], image[inverse[S],
              intersection[PO, complement[image[inverse[PS], intersection[PO, P[cart[x, x]]]]],
               P[cart[x, x]]]]]]], y → x] // Reverse

Out[53]= or[not[axch], not[member[x, V]],
           subclass[intersection[PO, P[cart[x, x]]], image[inverse[S],
             intersection[PO, complement[image[inverse[PS], intersection[PO, P[cart[x, x]]]]],
              P[cart[x, x]]]]]]] == True

In[54]:= (% /. x → x_) /. Equal → SetDelayed
```

The following lemma introduces the class **TO** of total orders.

```
In[55]:= SubstTest[implies, and[subclass[u, image[inverse[S], v]], subclass[v, w]],
           subclass[u, image[inverse[S], w]], {u -> intersection[PO, P[cart[x, x]]],
            v -> maximal[S, intersection[PO, P[cart[x, x]]]], w → TO}] // Reverse

Out[55]= or[not[subclass[intersection[PO, P[cart[x, x]]], image[inverse[S],
              intersection[PO, complement[image[inverse[PS], intersection[PO, P[cart[x, x]]]]]],
               P[cart[x, x]]]]]], not[subclass[intersection[PO, P[cart[x, x]]],
             union[TO, image[inverse[PS], intersection[PO, P[cart[x, x]]]]]]]],
           subclass[intersection[PO, P[cart[x, x]]], image[inverse[S], TO]]] == True

In[56]:= (% /. x → x_) /. Equal → SetDelayed
```

At this point one obtains the following consequence of Zorn's lemma: if the axiom of choice holds, then any partial order contained in the cartesian square of a set can be extended to a total order. It remains only to eliminate the set variable **x**.

```
In[57]:= Map[not, SubstTest[and, implies[and[p0, p1], p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[and[p0, p1], p4]],
            {p0 → axch, p1 → member[x, V], p2 -> subclass[intersection[PO, P[cart[x, x]]],
               image[inverse[S], maximal[S, intersection[PO, P[cart[x, x]]]]]],
             p3 -> subclass[maximal[S, intersection[PO, P[cart[x, x]]]], TO],
             p4 -> subclass[intersection[PO, P[cart[x, x]]], image[inverse[S], TO]]}]] //
         Reverse

Out[57]= or[not[axch], not[member[x, V]],
           subclass[intersection[PO, P[cart[x, x]]], image[inverse[S], TO]]] == True

In[58]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[59]:= SubstTest[implies, and[member[x, y], subclass[y, z]], member[x, z],
           {y → intersection[PO, P[cart[fix[x], fix[x]]]], z → image[inverse[S], TO]}] // Reverse

Out[59]= or[member[x, image[inverse[S], TO]], not[member[x, V]], not[PARTIALORDER[x]], not[
            subclass[intersection[PO, P[cart[fix[x], fix[x]]]], image[inverse[S], TO]]]] == True

In[60]:= (% /. x → x_) /. Equal → SetDelayed
```

The following reformulation facilitates the removal of the variable **x**.

```
In[61]:= Map[not, SubstTest[and, implies[and[p1, p3], p4],
            implies[p2, p3], implies[and[p2, p4], p5], not[implies[and[p1, p2], p5]],
            {p1 → axch, p2 → member[x, PO], p3 → member[fix[x], V],
             p4 -> subclass[intersection[PO, P[cart[fix[x], fix[x]]]], image[inverse[S], TO]],
             p5 → member[x, image[inverse[S], TO]]}]] // Reverse

Out[61]= or[member[x, image[inverse[S], TO]],
           not[axch], not[member[x, V]], not[PARTIALORDER[x]]] == True

In[62]:= or[member[x_, image[inverse[S], TO]],
           not[axch], not[member[x_, V]], not[PARTIALORDER[x_]]] := True
```

Szpilrajn Theorem. The axiom of choice implies that any partial order can be extended to a total order.

```
In[63]:= Map[equal[V, #] &,
          SubstTest[class, x, or[not[equal[t, V]], not[member[x, u]], member[x, v]],
           {t → SELECT, u → PO, v → image[inverse[S], TO]}]]

Out[63]= or[not[axch], subclass[PO, image[inverse[S], TO]]] == True

In[64]:= or[not[axch], subclass[PO, image[inverse[S], TO]]] := True
```