maximal partial orders in a cartesian square are total

Johan G. F. Belinfante
2009 May 1

```
In[1]:= SetDirectory["l:"]; << goedel.09apr30a;<< tools.m
```

```
:Package Title: goedel.09apr30a            2009 April 30 at 3:40 p.m.

It is now:  2009 May 1 at 4:33

Loading Simplification Rules

TOOLS.M                                    Revised 2009 April 6

weightlimit = 40
```

introduction

A maximal partial order contained in a fixed cartesian square is a total order.

adjoining a cartesian product

If **x** is a partial order and if an ordered pair **⟨u,v⟩** does not belong to **x**, 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**. A partial order cannot be maximal if it can be extended by adjoining a cartesian product. Thus if a partial order **x** is maximal in a cartesian square **cart[y, y]**, it must be a total order. One must 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[2]:= SubstTest[TRANSITIVE,
          union[trv[t], cart[image[inverse[trv[t]], u], image[trv[t], v]]], t → po[x]] // Reverse
```

```
Out[2]= TRANSITIVE[union[cart[image[inverse[po[x]], u], image[po[x], v]], po[x]]] == True
```

```
In[3]:= (% /. {u → u_, v → v_, x → x_}) /. Equal→ SetDelayed
```

Lemma.

```
In[4]:= SubstTest[REFLEXIVE, union[id[x], rfx[t]], t → po[y]] // Reverse
```

```
Out[4]= REFLEXIVE[union[id[x], po[y]]] == True
```

```
In[5]:= (% /. {x → x_, y → y_}) /. Equal→ SetDelayed
```

Lemma. For the reflexive property, the following suffices.

```
In[6]:= SubstTest[subclass, t, cartsq[fix[t]],
          t -> union[cart[image[inverse[po[x]], y], image[po[x], z]], po[x]]]

Out[6]= REFLEXIVE[union[cart[image[inverse[po[x]], y], image[po[x], z]], po[x]]] == True

In[7]:= (% /. {x → x_, y → y_, z → z_}) /. Equal→ SetDelayed
```

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

```
In[8]:= 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[8]= 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[9]:= (% /. {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[10]:= 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[10]= 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[11]:= (% /. {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[12]:= 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[12]= 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[13]:= (% /. {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[14]:= (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[14]= 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[15]:= (% /. {t → t_, y → y_, z → z_}) /. Equal → SetDelayed
```

Removing the **setpart** wrapper yields:

```
In[16]:= 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[16]= 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[17]:= (% /. {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[18]:= 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[18]= 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[19]:= (% /. {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[20]:= (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[20]= 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[21]:= (% /. t → t_) /. Equal -> SetDelayed
```

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

```
In[22]:= 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[22]= 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[23]:= (% /. x → x_) /. Equal → SetDelayed
```

---

adjoining an identity

In this section it is shown that if a partial order **x** is maximal in a cartesian square **cart[y, y]**, then **y** = **fix[x]**. The idea here is that a partial order cannot be maximal if it can be extended by adjoining an identity. We begin by using **po** wrappers, but these need to be removed in order to eliminate the variables.

Lemma. The desired equation **y = fix[po[x]]** is here replaced by an equivalent inclusion. 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[24]:= (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[24]= 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[25]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

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

```
In[26]:= 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[26]= 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[27]:= (% /. {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[28]:= 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[28]= 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[29]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[30]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
            not[implies[p1, p3]], {p1 → and[member[x, V], member[y, V],
               not[member[x, image[inverse[PS], intersection[PO, P[cart[y, y]]]]]],
               PARTIALORDER[x], subclass[x, cart[y, y]]],
             p2 → equal[y, fix[x]], p3 → TOTALORDER[x]}]] // Reverse

Out[30]= 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[31]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Eliminate a variable and introduce a **setpart** wrapper.

```
In[32]:= 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}]] /. y → setpart[x]

Out[32]= subclass[intersection[PO, P[cart[setpart[x], setpart[x]]]], union[TO,
           image[inverse[PS], intersection[PO, P[cart[setpart[x], setpart[x]]]]]]] == True

In[33]:= subclass[intersection[PO, P[cart[setpart[x_], setpart[x_]]]], union[TO,
           image[inverse[PS], intersection[PO, P[cart[setpart[x_], setpart[x_]]]]]]] := True
```

Restatement.

```
In[34]:= subclass[maximal[S, intersection[PO, P[cart[setpart[x], setpart[x]]]]], TO]

Out[34]= True
```

A variable-free restatement is possible, but at first appears to be rather complicated:

```
In[53]:= Map[empty[range[complement[#]]] &, SubstTest[reify, x,
           complement[dif[maximal[s, intersection[p, P[cart[setpart[x], setpart[x]]]]], t]],
           {p → PO, s → S, t → TO}]] // InvertFix

Out[53]= subclass[intersection[PO, fix[composite[inverse[S],
             id[image[CART, Id]], complement[composite[S, id[PO], PS]]]]], TO] == True

In[54]:= % /. Equal → SetDelayed
```

This can be simplified as follows. First replace **image[CART,Id]** with a variable **t**, and abstract on that.

```
In[48]:= abstract[t, intersection[PO,
           fix[composite[inverse[S], id[t], complement[composite[S, id[PO], PS]]]]]]

Out[48]= composite[inverse[E], DIF, id[composite[IMAGE[id[PO]], IMAGE[inverse[PS]]],
          inverse[FIRST], IMAGE[id[PO]], POWER]
```

The image of this is:

```
In[49]:= image[composite[inverse[E], DIF, id[composite[IMAGE[id[PO]], IMAGE[inverse[PS]]],
           inverse[FIRST], IMAGE[id[PO]], POWER], t]

Out[49]= image[MAXIMAL[S], image[IMAGE[id[PO]], image[POWER, t]]]
```

The above heuristics finally lead to this fairly nice variable-free formulation of the theorem.

Theorem.

```
In[62]:= (Map[subclass[image[#, y], z] &,
            composite[MAXIMAL[S], IMAGE[id[x]], POWER] // RelnNormality] /.
           {x → PO, y → image[CART, Id], z → TO}) // InvertFix

Out[62]= subclass[image[MAXIMAL[S], image[IMAGE[id[PO]], image[POWER, image[CART, Id]]]], TO] ==
          True

In[63]:= subclass[
           image[MAXIMAL[S], image[IMAGE[id[PO]], image[POWER, image[CART, Id]]]], TO] := True
```