PO and BIJ

Johan G. F. Belinfante
2004 December 18

```
In[1]:= SetDirectory["i:"]; << goedel64.18b; << tools.m
        :Package Title: goedel64.18b              2004 December 18 at 7:20 p.m.
        It is now:  2004 Dec 18 at 23:38
        Loading Simplification Rules
        TOOLS.M                       Revised 2004 December 16
        weightlimit = 40
```

summary

The class of small partial order relations satisfies an equation analogous to one for the class of small equivalence relations.

a basic observation

The first important observation is that to every bijection one can associate a partial order relation.

```
In[2]:= PARTIALORDER[composite[inverse[oopart[x]], inverse[S], oopart[x]]] //
         AssertTest
Out[2]= PARTIALORDER[composite[inverse[oopart[x]], inverse[S], oopart[x]]] == True
In[3]:= PARTIALORDER[composite[inverse[oopart[x_]], inverse[S], oopart[x_]]] := True
```

The following related result will not be needed further.

```
In[4]:= PARTIALORDER[composite[inverse[funpart[x]], inverse[S], funpart[x]]] //
         AssertTest
Out[4]= PARTIALORDER[composite[inverse[funpart[x]], inverse[S], funpart[x]]] ==
         FUNCTION[inverse[funpart[x]]]
In[5]:= PARTIALORDER[composite[inverse[funpart[x_]], inverse[S], funpart[x_]]] :=
         FUNCTION[inverse[funpart[x]]]
```

a variable-free reformulation

A variable-free restatement of the fact that **composite[inverse[oopart[x]], inverse[S], oopart[x]]** is a partial order can be obtained as follows.

```
In[6]:= Map[equal[V, #] &,
          SubstTest[class, x, subclass[image[u, singleton[setpart[x]]], v],
           {u -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
               composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]],
              OOPART], v → PO}]] // Reverse

Out[6]= subclass[BIJ, fix[composite[INVERSE, inverse[image[inverse[COMPOSE], PO]],
           IMAGE[cross[Id, inverse[S]]]]]] == True

In[7]:= % /. Equal → SetDelayed
```

The following fact was discovered in an earlier notebook concerned with quasi-orders.

```
In[8]:= SubstTest[composite, funpart[x], inverse[funpart[x]],
         x -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
            composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]]]

Out[8]= composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
           composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]],
          intersection[composite[INVERSE, FIRST], composite[
            inverse[IMAGE[cross[Id, inverse[S]]]], SECOND]], inverse[COMPOSE]] ==
         id[image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE]]]

In[9]:= % /. Equal → SetDelayed
```

Corollary:

```
In[10]:= ImageComp[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
             composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]],
           inverse[composite[COMPOSE, intersection[
              composite[inverse[FIRST], INVERSE], composite[inverse[SECOND],
               IMAGE[cross[Id, inverse[S]]]]]]], x] // Reverse

Out[10]= image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]],
            id[fix[composite[INVERSE, inverse[image[inverse[COMPOSE], x]],
               IMAGE[cross[Id, inverse[S]]]]], INVERSE]] == intersection[x,
          image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE]]]
```

This is made into a temporary rewrite rule:

```
In[11]:= image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]],
           id[fix[composite[INVERSE, inverse[image[inverse[COMPOSE], x_]],
              IMAGE[cross[Id, inverse[S]]]]]], INVERSE]] := intersection[x,
          image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE]]]
```

The statement obtained above is then transformed as follows:

```
In[12]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → BIJ, v -> fix[composite[INVERSE,
              inverse[image[inverse[COMPOSE], PO]], IMAGE[cross[Id, inverse[S]]]]],
           w -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]]}]

Out[12]= subclass[image[COMPOSE,
           composite[IMAGE[cross[Id, inverse[S]]], INVERSE, id[BIJ]]], PO] == True

In[13]:= % /. Equal → SetDelayed
```

A reverse inclusion will be derived in a later section.

---

lemma

The following general fact is used later.

```
In[14]:= SubstTest[subclass, composite[inverse[z], z], Id, z → composite[x, id[y]]]

Out[14]= subclass[composite[id[y], inverse[x], x, id[y]], Id] ==
          FUNCTION[composite[id[y], inverse[x]]]

In[15]:= subclass[composite[id[y_], inverse[x_], x_, id[y_]], Id] :=
          FUNCTION[composite[id[y], inverse[x]]]
```

---

one-to-one property

In this section it is shown that the natural map for a partial order is one-to-one. The starting point uses an argument that takes advantage of a triple wrapper for a generic small reflexive transitive relation.

```
In[16]:= Map[implies[ANTISYMMETRIC[y], subclass[#, Id]] &,
           (composite[id[fix[y]], inverse[VERTSECT[y]], VERTSECT[y], id[fix[y]]] //
             RelnNormality)] /. y → rfx[trv[setpart[x]]]

Out[16]= or[FUNCTION[composite[id[fix[trv[setpart[x]]]],
            inverse[VERTSECT[rfx[trv[setpart[x]]]]]],
           not[subclass[intersection[inverse[rfx[trv[setpart[x]]]],
              rfx[trv[setpart[x]]]], Id]]] == True

In[17]:= (% /. x → x_) /. Equal → SetDelayed
```

The wrappers are removed as follows:

```
In[18]:= SubstTest[implies, and[ANTISYMMETRIC[y], equal[y, rfx[trv[setpart[x]]]]],
          FUNCTION[inverse[composite[VERTSECT[y], id[domain[y]]]], y → setpart[x]]

Out[18]= or[FUNCTION[
            composite[id[domain[setpart[x]]], inverse[VERTSECT[setpart[x]]]],
           not[PARTIALORDER[setpart[x]]]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

The variable **x** is eliminated as follows:

```
In[20]:= Map[equal[V, #] &, SubstTest[class, x, implies[
             member[setpart[x], u], subclass[image[v, singleton[setpart[x]]], w]],
            {u → PO, v → VS, w → BIJ}]] // Reverse

Out[20]= subclass[image[VS, PO], BIJ] == True

In[21]:= subclass[image[VS, PO], BIJ] := True
```

---

an antitone property

Some of the material in this section repeats material in a notebook about reflexive transitive relations. Recall that for any thin reflexive transitive relation **x** there is a canonical factorization of the form **x = composite[inverse[funpart[y]], inverse[S], funpart[y]]** where **y** is the natural map, that is, the restriction of the vertical section function to **fix[x]**. For sets, this result is:

```
In[22]:= (equal[z, composite[inverse[funpart[y]], inverse[S], funpart[y]]] /.
            y -> composite[VERTSECT[z], id[fix[z]]]) /. z -> rfx[trv[setpart[x]]]

Out[22]= True
```

The variable **x** can be eliminated as follows.

```
In[23]:= SubstTest[class, x,
           equal[image[u, singleton[setpart[x]]], image[v, singleton[setpart[x]]]],
           {u -> composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
                composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]],
              IMAGE[id[cart[V, V]]], VS, CORE[RFX], HULL[TRV], IMAGE[id[cart[V, V]]]],
            v → composite[CORE[RFX], HULL[TRV], IMAGE[id[cart[V, V]]]]}] // Reverse

Out[23]= image[inverse[IMAGE[id[cart[V, V]]]],
           image[inverse[HULL[TRV]], image[inverse[CORE[RFX]],
             fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
                composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]], VS]]]] == V

In[24]:= % /. Equal → SetDelayed
```

The wrapper-related functions **CORE[RFX]** and **composite[HULL[TRV], IMAGE[id[cart[V,V]]]]** are eliminated using **ImageComp**:

```
In[25]:= Map[subclass[intersection[RFX, TRV], #] &,
          ImageComp[composite[CORE[RFX], HULL[TRV], IMAGE[id[cart[V, V]]]],
           inverse[composite[CORE[RFX], HULL[TRV], IMAGE[id[cart[V, V]]]]],
           fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]], VS]]]]

Out[25]= subclass[intersection[RFX, TRV],
           fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]], VS]]] == True

In[26]:= % /. Equal → SetDelayed
```

Since any partial order is reflexive and transitive, one obtains this corollary:

```
In[27]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → PO, v -> intersection[RFX, TRV],
           w -> fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
                composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]], VS]]}]

Out[27]= subclass[PO,
           fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]], VS]]] == True

In[28]:= % /. Equal → SetDelayed
```

derivation of the main result

The trick now is to combine the inclusion found in the preceding section with the fact **subclass[image[VS,PO], BIJ]**. The key to doing so is to rewrite this fact as follows:

```
In[29]:= subclass[PO, fix[composite[inverse[cart[V, BIJ]], VS]]]

Out[29]= True
```

The desired combination is achieved as follows:

```
In[30]:= Map[subclass[PO, #] &,
          SubstTest[fix, composite[intersection[u, v], funpart[w]], {u → cart[BIJ, V],
            v → composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
               composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]]], w → VS}]]

Out[30]= subclass[PO, fix[composite[COMPOSE,
             intersection[composite[inverse[FIRST], INVERSE], composite[
               inverse[SECOND], IMAGE[cross[Id, inverse[S]]]], id[BIJ], VS]]] == True

In[31]:= % /. Equal → SetDelayed
```

The next step needs this relation, which was also used in our earlier study of quasi-orders:

```
In[32]:= abstract[x,
          fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
             composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]], x]]]

Out[32]= composite[FIRST,
          id[composite[intersection[composite[INVERSE, FIRST], composite[
              inverse[IMAGE[cross[Id, inverse[S]]], SECOND]], inverse[COMPOSE]]]]
```

The idea is to eliminate **VS** by using the following inclusion:

```
In[33]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → composite[id[BIJ], VS], v → cart[V, BIJ], w -> composite[FIRST,
             id[composite[intersection[composite[INVERSE, FIRST], composite[inverse[
                  IMAGE[cross[Id, inverse[S]]], SECOND]], inverse[COMPOSE]]]]}]

Out[33]= subclass[
           fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
              composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]], id[BIJ], VS]],
           image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]],
             INVERSE, id[BIJ]]]] == True
```

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

Combining this inclusion with the one derived for **PO** yields:

```
In[35]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → PO,
           v -> fix[composite[COMPOSE, intersection[composite[inverse[FIRST], INVERSE],
               composite[inverse[SECOND], IMAGE[cross[Id, inverse[S]]]]],
              id[BIJ], VS]], w -> image[COMPOSE,
            composite[IMAGE[cross[Id, inverse[S]]], INVERSE, id[BIJ]]]}]
```

```
Out[35]= subclass[PO, image[COMPOSE,
           composite[IMAGE[cross[Id, inverse[S]]], INVERSE, id[BIJ]]]] == True
```

```
In[36]:= % /. Equal → SetDelayed
```

The reverse inclusion was proved in an earlier section. Combining these yields a variable-free equation:

```
In[37]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → PO, v ->
             image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE, id[BIJ]]]}]
```

```
Out[37]= True == equal[PO,
           image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE, id[BIJ]]]]
```

```
In[38]:= image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], INVERSE, id[BIJ]]] := PO
```

Note the close resemblance between this formula and a previously derived formula for the class of quasi-orders:

```
In[39]:= image[COMPOSE, composite[IMAGE[cross[Id, inverse[S]]], id[FUNS], INVERSE]]
```

```
Out[39]= intersection[RFX, TRV]
```