transforms of partial orders

Johan G. F. Belinfante
2006 January 25

```
In[1]:= SetDirectory["l:"]; << goedel77.25a; << tools.m

        :Package Title: goedel77.25a             2006 January 25 at 9:15 p.m.

        It is now:  2006 Jan 25 at 21:17

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 January 2

        weightlimit = 40
```

summary

Every partial order is order-isomorphic to a restriction of **inverse[S]**. A variable-free formulation of this is derived and used to express the class of partial orders as the class of transforms of **inverse[S]** by bijections.

transforms of partial orders

The following rewrite rule resembles an existing one, except for interchanging **oopart[x]** with its inverse. It says that transforming a partial order with a bijection yields another partial order. Comment: the transformed partial order need not be isomorphic to the original one. For example, the bijection could be a restriction of the identity map, yielding a restriction of the original partial order.

```
In[2]:= SubstTest[PARTIALORDER, composite[oopart[z], po[y], inverse[oopart[z]]], z → inverse[x]]

Out[2]= PARTIALORDER[composite[inverse[oopart[x]], po[y], oopart[x]]] == True

In[3]:= PARTIALORDER[composite[inverse[oopart[x_]], po[y_], oopart[x_]]] := True
```

Removing the **oopart** wrappers yields:

```
In[4]:= SubstTest[implies, equal[y, oopart[z]],
         PARTIALORDER[composite[inverse[y], po[x], y]], z → y]

Out[4]= or[not[FUNCTION[y]], not[FUNCTION[inverse[y]]],
          PARTIALORDER[composite[inverse[y], po[x], y]]] == True

In[5]:= or[not[FUNCTION[y_]], not[FUNCTION[inverse[y_]]],
          PARTIALORDER[composite[inverse[y_], po[x_], y_]]] := True
```

Double negation yields the following temporary lemma needed for the next step.

```
In[10]:= or[and[member[image[inverse[z], po[x]], V], PARTIALORDER[image[inverse[z], po[x]]]],
           not[equal[z, cross[y, y]]], not[FUNCTION[y]],
           not[FUNCTION[inverse[y]]], not[member[y, V]]] // NotNotTest

Out[10]= or[and[member[image[inverse[z], po[x]], V], PARTIALORDER[image[inverse[z], po[x]]]],
           not[equal[z, cross[y, y]]], not[FUNCTION[y]],
           not[FUNCTION[inverse[y]]], not[member[y, V]]] == True

In[11]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

First an extra variable **z** is introduced, and then both of the variables **y** and **z** are eliminated:

```
In[12]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[y, z], implies[and[equal[z, cross[y, y]], member[y, u]],
             member[image[inverse[z], w], v]], {u → BIJ, v → PO, w → po[x]}]] // Reverse

Out[12]= subclass[image[IMAGE[FIRST],
            image[IMAGE[id[cart[V, po[x]]]], image[CROSS, id[BIJ]]], PO] == True

In[13]:= subclass[image[IMAGE[FIRST],
            image[IMAGE[id[cart[V, po[x_]]]], image[CROSS, id[BIJ]]], PO] := True
```

Dual result.

```
In[14]:= Map[subclass[#, PO] &, ImageComp[composite[IMAGE[FIRST], IMAGE[id[cart[V, po[x]]]]],
           IMAGE[SWAP], image[CROSS, id[BIJ]]]]

Out[14]= subclass[image[IMAGE[SECOND],
            image[IMAGE[id[cart[po[x], V]]], image[CROSS, id[BIJ]]], PO] == True

In[15]:= subclass[image[IMAGE[SECOND],
            image[IMAGE[id[cart[po[x_], V]]], image[CROSS, id[BIJ]]], PO] := True
```

We note the following special case of interest. The following inclusion will later be sharpened to an equation.

```
In[16]:= SubstTest[subclass, image[IMAGE[FIRST],
           image[IMAGE[id[cart[V, po[x]]]], image[CROSS, id[BIJ]]], PO, x → inverse[S]]

Out[16]= subclass[image[IMAGE[FIRST],
            image[IMAGE[id[cart[V, inverse[S]]]], image[CROSS, id[BIJ]]], PO] == True

In[17]:= % /. Equal → SetDelayed
```

applying the theory of the natural map

The key to deriving the resverse inclusion alluded to above is to make use of the theory of the natural mapping for partial orders. It is convenient to introduce an extra variable **y** which will later be eliminated.

```
In[18]:= SubstTest[implies,
          and[equal[x, po[z]], equal[y, composite[VERTSECT[x], id[domain[x]]]]],
          equal[thinpart[x], composite[inverse[y], inverse[S], y]], z → x]

Out[18]= or[equal[composite[inverse[y], inverse[S], y], thinpart[x]],
           not[equal[y, composite[VERTSECT[x], id[domain[x]]]]], not[PARTIALORDER[x]]] == True

In[19]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

If **x** is a set, the **thinpart** wrapper is not needed:

```
In[20]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[p1, p4], implies[p1, p5], implies[and[p3, p4, p5], p6],
           not[implies[and[p1, p2], p6]], {p1 → member[x, PO], p2 → member[pair[x, y], VS],
            p3 -> equal[thinpart[x], composite[inverse[y], inverse[S], y]],
            p4 → thin[x], p5 → subclass[x, cart[V, V]],
            p6 -> equal[x, composite[inverse[y], inverse[S], y]]}]]

Out[20]= or[equal[x, composite[inverse[y], inverse[S], y]],
           not[equal[y, composite[VERTSECT[x], id[domain[x]]]]],
           not[member[x, V]], not[PARTIALORDER[x]]] == True

In[21]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[22]:= equiv[and[member[y, V], member[cross[y, y], V]], member[y, V]]

Out[22]= True

In[23]:= and[member[y_, V], member[cross[y_, y_], V]] := member[y, V]
```

Main theorem about the natural mapping, in variable-free form.

```
In[24]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[x, y], implies[and[member[x, u], member[y, V],
              member[pair[x, y], v]], member[pair[y, x], w]], {u → PO, v → VS, w ->
              composite[IMAGE[FIRST], IMAGE[id[cart[V, inverse[S]]]], CROSS, DUP]}]] // Reverse

Out[24]= subclass[PO,
           fix[composite[IMAGE[FIRST], IMAGE[id[cart[V, inverse[S]]]], CROSS, DUP, VS]]] == True

In[25]:= subclass[PO,
           fix[composite[IMAGE[FIRST], IMAGE[id[cart[V, inverse[S]]]], CROSS, DUP, VS]]] := True
```

application

To apply the result of the preceding section, the following corollary is needed.

```
In[26]:= equal[intersection[PO,
           fix[composite[IMAGE[FIRST], IMAGE[id[cart[V, inverse[S]]]], CROSS, DUP, VS]]], PO]

Out[26]= True

In[27]:= intersection[PO,
           fix[composite[IMAGE[FIRST], IMAGE[id[cart[V, inverse[S]]]], CROSS, DUP, VS]]] := PO
```

The natural mappings for partial orders are one-to-one. This implies that **composite[VS, id[PO]]** is contained in **cart[PO, BIJ]**.

```
In[28]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> composite[VS, id[PO]], v -> cart[PO, BIJ],
           w → composite[FIRST, id[composite[inverse[DUP], inverse[CROSS],
               inverse[IMAGE[id[cart[V, inverse[S]]]]], inverse[IMAGE[FIRST]]]]]}]

Out[28]= subclass[PO, image[IMAGE[FIRST],
            image[IMAGE[id[cart[V, inverse[S]]]], image[CROSS, id[BIJ]]]]] == True

In[29]:= % /. Equal → SetDelayed
```

Final result.

```
In[30]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → PO, v ->
            image[IMAGE[FIRST], image[IMAGE[id[cart[V, inverse[S]]]], image[CROSS, id[BIJ]]]]}]

Out[30]= True == equal[PO,
           image[IMAGE[FIRST], image[IMAGE[id[cart[V, inverse[S]]]], image[CROSS, id[BIJ]]]]]

In[31]:= image[IMAGE[FIRST], image[IMAGE[id[cart[V, inverse[S]]]], image[CROSS, id[BIJ]]]] := PO
```

Dual result:

```
In[32]:= ImageComp[composite[IMAGE[FIRST], IMAGE[id[cart[V, inverse[S]]]]],
          IMAGE[SWAP], image[CROSS, id[BIJ]]]

Out[32]= image[IMAGE[SECOND], image[IMAGE[id[cart[inverse[S], V]]], image[CROSS, id[BIJ]]]] == PO

In[33]:= image[IMAGE[SECOND],
           image[IMAGE[id[cart[inverse[S], V]]], image[CROSS, id[BIJ]]]] := PO
```