from partial order to strict order and back

Johan G. F. Belinfante
2009 June 16

```
In[1]:= SetDirectory["l:"]; << goedel.09jun14a; << tools.m

        :Package Title: goedel.09jun14a              2009 June 14 at 7:15 a.m.

        It is now:  2009 Jun 16 at 17:17

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 June 1

        weightlimit = 40
```

summary

From a partial order **po[x]** one obtains a strict order **Di ⋂ po[x]**, and by taking its reflexive hull, one gets back a partial order, but not necessarily the same that one started with. The class of reflexive hulls of strict orders is a subclass of the class of all partial orders. In this notebook it is shown that this class is precisely the class of all partial orders without elements that are both minimal and maximal at the same time.

derivation

Lemma.

```
In[2]:= SubstTest[hull, RFX, composite[Id, setpart[t]],
          t -> intersection[Di, po[setpart[x]]]] // Reverse

Out[2]= hull[RFX, intersection[Di, po[setpart[x]]]] ==
         intersection[complement[oopart[po[setpart[x]]]], po[setpart[x]]]

In[3]:= hull[RFX, intersection[Di, po[setpart[x_]]]] :=
         intersection[complement[oopart[po[setpart[x]]]], po[setpart[x]]]
```

Lemma.

```
In[4]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]], {t → HULL[RFX],
          u → set[intersection[Di, po[setpart[x]]]], v → intersection[TRV, P[Di]]}] // Reverse

Out[4]= member[intersection[complement[oopart[po[setpart[x]]]], po[setpart[x]]],
          image[HULL[RFX], intersection[TRV, P[Di]]]] == True

In[5]:= member[intersection[complement[oopart[po[setpart[x_]]]], po[setpart[x_]]],
          image[HULL[RFX], intersection[TRV, P[Di]]]] := True
```

Theorem.

```
In[6]:= SubstTest[implies, equal[t, oopart[po[setpart[x]]]],
          member[intersection[complement[t], po[setpart[x]]],
           image[HULL[RFX], intersection[TRV, P[Di]]]], t → 0] // Reverse

Out[6]= or[member[po[setpart[x]], image[HULL[RFX], intersection[TRV, P[Di]]]],
          not[equal[0, oopart[po[setpart[x]]]]]] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary. (Obtained by removing the **po[setpart[x]]** compound wrapper.)

```
In[8]:= SubstTest[implies, equal[x, po[setpart[t]]],
          or[member[x, image[HULL[RFX], intersection[TRV, P[Di]]]],
           not[equal[0, oopart[x]]]], t → x] // Reverse

Out[8]= or[member[x, image[HULL[RFX], intersection[TRV, P[Di]]]],
          not[equal[0, oopart[x]]], not[member[x, V]], not[PARTIALORDER[x]]] == True

In[9]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[10]:= member[x, dif[intersection[PO, image[inverse[OOPART], set[0]]],
            image[HULL[RFX], intersection[TRV, P[Di]]]]] // NotNotTest

Out[10]= and[equal[0, oopart[x]], member[x, V],
           not[member[x, image[HULL[RFX], intersection[TRV, P[Di]]]]], PARTIALORDER[x]] == False

In[11]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (Obtained by eliminating the variable **x**.)

```
In[12]:= Map[empty, SubstTest[class, x, member[x, t],
           t -> dif[intersection[PO, image[inverse[OOPART], set[0]]],
             image[HULL[RFX], intersection[TRV, P[Di]]]]]]

Out[12]= subclass[intersection[PO, image[inverse[OOPART], set[0]]],
           image[HULL[RFX], intersection[TRV, P[Di]]]] == True

In[13]:= % /. Equal → SetDelayed
```

The **simplify** flag needs to be cleared to prevent looping for the next result.

```
In[14]:= simplify = False;
```

Lemma.

```
In[15]:= SubstTest[funpart, composite[s, id[t]],
           {s → po[x], t → complement[fix[oopart[po[x]]]]}] // Reverse

Out[15]= funpart[intersection[complement[oopart[po[x]]], po[x]]] ==
          composite[funpart[po[x]], id[complement[fix[oopart[po[x]]]]]]
```

```
In[16]:= funpart[intersection[complement[oopart[po[x_]]], po[x_]]] :=
          composite[funpart[po[x]], id[complement[fix[oopart[po[x]]]]]]
```

Theorem.

```
In[17]:= Map[empty, (oopart[composite[id[t], s]] // RelnNormality) /.
           {s → po[x], t → complement[fix[oopart[po[x]]]]}]
Out[17]= equal[0, oopart[intersection[complement[oopart[po[x]]], po[x]]]] == True

In[18]:= oopart[intersection[complement[oopart[po[x_]]], po[x_]]] := 0
```

The variable **x** can be eliminated using **reify**.

Theorem.

```
In[19]:= Map[empty[composite[#, id[PO]]] &,
          SubstTest[reify, x, oopart[hull[RFX, intersection[t, po[setpart[x]]]]], t → Di]]
Out[19]= subclass[image[OOPART, image[HULL[RFX], intersection[TRV, P[Di]]]], set[0]] == True

In[20]:= % /. Equal→ SetDelayed

In[21]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[HULL[RFX], intersection[TRV, P[Di]]],
            v → intersection[PO, image[inverse[OOPART], set[0]]]}]
Out[21]= equal[image[HULL[RFX], intersection[TRV, P[Di]]],
           intersection[PO, image[inverse[OOPART], set[0]]]] == True

In[22]:= image[HULL[RFX], intersection[TRV, P[Di]]] :=
          intersection[PO, image[inverse[OOPART], set[0]]]
```

---

a difference formula

In this section a variable-free rewrite rule is derived corresponding to this known rule:

```
In[23]:= union[intersection[po[x], Di], id[udora[intersection[po[x], Di]]]]
Out[23]= intersection[complement[oopart[po[x]]], po[x]]
```

Lemma. Simplification rule.

```
In[24]:= composite[inverse[E], DIF, id[composite[OOPART, id[PO]]], inverse[FIRST]] //
           ReifNormality// Reverse
Out[24]= composite[intersection[composite[complement[inverse[E]], OOPART], inverse[E]],
           id[PO]] == composite[inverse[E], DIF, id[composite[OOPART, id[PO]]], inverse[FIRST]]

In[25]:= % /. Equal→ SetDelayed
```

Lemma. Simplification rule.

```
In[26]:= Assoc[intersection[composite[complement[inverse[E]], OOPART],
           composite[inverse[E], IMAGE[id[cart[V, V]]]]], id[P[cart[V, V]]], id[PO]]

Out[26]= composite[intersection[composite[complement[inverse[E]], OOPART],
           composite[inverse[E], IMAGE[id[cart[V, V]]]]], id[PO]] ==
         composite[inverse[E], DIF, id[composite[OOPART, id[PO]]], inverse[FIRST]]

In[27]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[28]:= Map[composite[VERTSECT[composite[#, id[P[cart[V, V]]]], id[PO]] &,
           SubstTest[reify, x, hull[t, intersection[Di, po[setpart[x]]], t→ RFX]] // Reverse

Out[28]= composite[DIF, id[composite[OOPART, id[PO]]], inverse[FIRST]] ==
         composite[HULL[RFX], IMAGE[id[Di]], id[PO]]

In[29]:= composite[DIF, id[composite[OOPART, id[PO]]], inverse[FIRST]] :=
         composite[HULL[RFX], IMAGE[id[Di]], id[PO]]
```

Corollary.

```
In[30]:= ImageComp[composite[DIF, id[composite[OOPART, id[PO]]]], inverse[FIRST], V] // Reverse

Out[30]= image[DIF, composite[OOPART, id[PO]]] ==
         intersection[PO, image[inverse[OOPART], set[0]]]

In[31]:= image[DIF, composite[OOPART, id[PO]]] :=
         intersection[PO, image[inverse[OOPART], set[0]]]
```

---

an example

Every identity function is a partial order, but only the empty identity function is the reflexive hull of a strict order.

Theorem.

```
In[32]:= Map[image[IDP, #] &, IminComp[OOPART, IDP, set[0]]] // Reverse

Out[32]= intersection[image[inverse[OOPART], set[0]], P[Id]] == set[0]

In[33]:= intersection[image[inverse[OOPART], set[0]], P[Id]] := set[0]
```