complete lattices

Johan G. F. Belinfante
2005 October 2

```
In[1]:= SetDirectory["l:"]; << goedel74.02b; << tools.m

        :Package Title: goedel74.02b            2005 October 2 at 7:15 p.m.

        It is now:  2005 Oct 2 at 19:52

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 October 2

        weightlimit = 40
```

summary

Every subset of a complete lattice has a least upper bound and a greatest lower bound. These two conditions are not independent. Each implies the other. These facts are derived in this notebook using **po** wrappers. Each of the equations **domain[GLB[po[x]]] = P[fix[po[x]]]** and **domain[LUB[po[x]]] = P[fix[po[x]]]** implies the other. The use of **po** wrappers to replace **PARTIALORDER** literals does not significantly shorten the reasoning, but it does help a little with duality arguments, and with automating some translations of standard results needed because of rewrite rules that cause the domain and range of a partial order relation to be rewritten to the set of its fixed points.

some general results

The results in this notebook only apply to the case of small partial order relations. A partial order relation is a set if and only if its fixed points form a set.

```
In[2]:= member[po[x], V] // AssertTest // Reverse

Out[2]= member[fix[po[x]], V] == member[po[x], V]

In[3]:= member[fix[po[x_]], V] := member[po[x], V]
```

The basic chain rule for inclusion must sometimes be supplemented with additional rewrite rules. A general case is this:

```
In[4]:= SubstTest[implies, and[subclass[w, v], subclass[v, z]],
         subclass[w, z], v → intersection[x, y]]

Out[4]= or[not[subclass[w, x]], not[subclass[w, y]],
          not[subclass[intersection[x, y], z]], subclass[w, z]] == True
```

```
In[5]:= or[not[subclass[w_, x_]], not[subclass[w_, y_]],
          not[subclass[intersection[x_, y_], z_]], subclass[w_, z_]] := True
```

A particular case of this will be needed below:

```
In[6]:= SubstTest[implies, and[subclass[y, u], subclass[y, v], subclass[intersection[u, v], w]],
         subclass[y, w], v → lb[po[x], z]]
```

```
Out[6]= or[not[subclass[y, u]], not[subclass[cart[y, z], po[x]]],
          not[subclass[intersection[u, lb[po[x], z]], w]], subclass[y, w]] == True
```

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

---

derivation

Lemma.

```
In[8]:= Map[not, SubstTest[and, implies[and[p2, p3, p4], p6], implies[p6, p7], implies[p7, p9],
          not[implies[and[p2, p3, p4], p9]], {p2 → equal[domain[GLB[po[x]]], P[fix[po[x]]]],
           p3 → member[y, V], p4 → subclass[y, fix[po[x]]], p6 → member[y, domain[GLB[po[x]]]],
           p7 → subclass[intersection[fix[po[x]], lb[po[x], y]],
             image[inverse[po[x]], set[APPLY[GLB[po[x]], y]]]],
           p9 → subclass[intersection[fix[po[x]], lb[po[x], y]],
             image[inverse[po[x]], set[APPLY[GLB[po[x]], y]]]]}]]
```

```
Out[8]= or[not[equal[domain[GLB[po[x]]], P[fix[po[x]]]]], not[member[y, V]],
          not[subclass[y, fix[po[x]]]], subclass[intersection[fix[po[x]], lb[po[x], y]],
           image[inverse[po[x]], set[APPLY[GLB[po[x]], y]]]]] == True
```

```
In[9]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

If **y** is a subset of **fix[po[x]]**, then so is **z = intersection[fix[po[x]], ub[po[x],y]]**. The preceding result can now be applied to **z**.

```
In[10]:= Map[not, SubstTest[and, implies[and[p0, p2, p3], p4], implies[p3, p5],
           implies[and[p2, p4, p5], p6], not[implies[and[p0, p2, p3], p6]],
           {p0 → member[po[x], V], p2 → equal[domain[GLB[po[x]]], P[fix[po[x]]]],
            p3 → equal[z, intersection[fix[po[x]], ub[po[x], y]]], p4 → member[z, V],
            p5 → subclass[z, fix[po[x]]], p6 → subclass[intersection[fix[po[x]], lb[po[x], z]],
              image[inverse[po[x]], set[APPLY[GLB[po[x]], z]]]]}]]
```

```
Out[10]= or[not[equal[z, intersection[fix[po[x]], ub[po[x], y]]]],
           not[equal[domain[GLB[po[x]]], P[fix[po[x]]]]], not[member[po[x], V]],
           subclass[intersection[fix[po[x]], lb[po[x], z]],
            image[inverse[po[x]], set[APPLY[GLB[po[x]], z]]]]] == True
```

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

Restatement.

```
In[12]:= implies[and[member[po[x], V], equal[domain[GLB[po[x]]], P[fix[po[x]]]],
             member[y, P[fix[po[x]]]], equal[z, intersection[fix[po[x]], ub[po[x], y]]]],
           member[pair[z, APPLY[GLB[po[x]], z]], GLB[po[x]]]] // not // not

Out[12]= True
```

Lemma.

```
In[13]:= Map[not, SubstTest[and, implies[and[p1, p3, p4, p5, p6], p7],
           implies[and[p1, p3, p4, p5, p6], p8], implies[and[p3, p7, p8], p9],
           implies[p6, p10], implies[and[p5, p9, p10], p11],
           not[implies[and[p1, p3, p4, p5, p6], p11]], {p1 → member[po[x], V],
            p3 → equal[domain[GLB[po[x]]], P[fix[po[x]]]], p4 → member[y, V],
            p5 → subclass[y, fix[po[x]]], p6 → equal[z, intersection[fix[po[x]], ub[po[x], y]]],
            p7 → member[z, V], p8 → subclass[z, fix[po[x]]],
            p9 → subclass[intersection[fix[po[x]], lb[po[x], z]], image[inverse[po[x]],
               set[APPLY[GLB[po[x]], z]]]], p10 → subclass[cart[y, z], po[x]],
            p11 → subclass[y, image[inverse[po[x]], set[APPLY[GLB[po[x]], z]]]]}]]

Out[13]= or[not[equal[z, intersection[fix[po[x]], ub[po[x], y]]]],
           not[equal[domain[GLB[po[x]]], P[fix[po[x]]]]], not[member[y, V]],
           not[member[po[x], V]], not[subclass[y, fix[po[x]]]],
           subclass[y, image[inverse[po[x]], set[APPLY[GLB[po[x]], z]]]]] == True

In[14]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[15]:= SubstTest[implies,
           and[member[y, V], member[w, least[po[x], intersection[fix[po[x]], ub[po[x], y]]]]],
           member[y, domain[LUB[po[x]]]], w → APPLY[GLB[po[x]], z]]

Out[15]= or[member[y, domain[LUB[po[x]]]],
           not[member[y, V]], not[member[z, domain[GLB[po[x]]]]],
           not[subclass[y, image[inverse[po[x]], set[APPLY[GLB[po[x]], z]]]]],
           not[subclass[intersection[fix[po[x]], ub[po[x], y]],
             image[po[x], set[APPLY[GLB[po[x]], z]]]]] == True

In[16]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The variable **z** can be eliminated at this point.

```
In[17]:= Map[not, SubstTest[and, implies[and[p1, p3, p4, p5, p6], p7],
            implies[and[p1, p3, p4, p5, p6], p8], implies[and[p1, p3, p4, p5, p6], p9],
            implies[and[p4, p7, p8, p9], p10], not[implies[and[p1, p3, p4, p5, p6], p10]],
            {p1 → member[po[x], V], p3 → equal[domain[GLB[po[x]]], P[fix[po[x]]]],
             p4 → member[y, V], p5 → subclass[y, fix[po[x]]],
             p6 → equal[z, intersection[fix[po[x]], ub[po[x], y]]],
             p7 → member[APPLY[GLB[po[x]], z], fix[po[x]]], p8 → subclass[
               intersection[fix[po[x]], ub[po[x], y]], image[po[x], set[APPLY[GLB[po[x]], z]]]],
             p9 → subclass[y, image[inverse[po[x]], set[APPLY[GLB[po[x]], z]]]],
             p10 → member[y, domain[LUB[po[x]]]]}]] /.
         z -> intersection[fix[po[x]], ub[po[x], y]]

Out[17]= or[member[y, domain[LUB[po[x]]]], not[equal[domain[GLB[po[x]]], P[fix[po[x]]]]],
           not[member[y, V]], not[member[po[x], V]], not[subclass[y, fix[po[x]]]]] == True

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

The variable **y** can also be eliminated:

```
In[19]:= Map[equal[V, #] &, SubstTest[class, y,
            implies[and[member[t, V], equal[v, z], member[y, z]], member[y, w]], {t → po[x],
             v → domain[GLB[po[x]]], w → domain[LUB[po[x]]], z → P[fix[po[x]]]}]] // Reverse

Out[19]= or[not[member[po[x], V]], not[subclass[P[fix[po[x]]], domain[GLB[po[x]]]]],
           subclass[P[fix[po[x]]], domain[LUB[po[x]]]]] == True

In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

This result can be cleaned up a bit, replacing **subclass** with **equal**:

```
In[21]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
           {p → and[member[po[x], V], subclass[P[fix[po[x]]], domain[GLB[po[x]]]]],
            u → P[fix[po[x]]], v → domain[LUB[po[x]]]}] // Reverse

Out[21]= or[equal[domain[LUB[po[x]]], P[fix[po[x]]]], not[member[po[x], V]],
           not[subclass[P[fix[po[x]]], domain[GLB[po[x]]]]]] == True

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

Main theorem:

```
In[23]:= Map[not, SubstTest[and, implies[p2, p3],
           implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → member[po[x], V], p2 → equal[domain[GLB[po[x]]], P[fix[po[x]]]],
            p3 → subclass[P[fix[po[x]]], domain[GLB[po[x]]]],
            p4 → equal[domain[LUB[po[x]]], P[fix[po[x]]]]}]]

Out[23]= or[equal[domain[LUB[po[x]]], P[fix[po[x]]]],
           not[equal[domain[GLB[po[x]]], P[fix[po[x]]]]], not[member[po[x], V]]] == True

In[24]:= or[equal[domain[LUB[po[x_]]], P[fix[po[x_]]]],
           not[equal[domain[GLB[po[x_]]], P[fix[po[x_]]]]], not[member[po[x_], V]]] := True
```

Dual theorem:

```
In[25]:= SubstTest[or, equal[domain[LUB[po[y]]], P[fix[po[y]]]],
          not[equal[domain[GLB[po[y]]], P[fix[po[y]]]]], not[member[po[y], V]], y → inverse[x]]

Out[25]= or[equal[domain[GLB[po[x]]], P[fix[po[x]]]],
           not[equal[domain[LUB[po[x]]], P[fix[po[x]]]]], not[member[po[x], V]]] == True

In[26]:= or[equal[domain[GLB[po[x_]]], P[fix[po[x_]]]],
           not[equal[domain[LUB[po[x_]]], P[fix[po[x_]]]]], not[member[po[x_], V]]] := True
```

restatement using the PARTIALORDER predicate

The results obtained can be restated using the **PARTIALORDER** predicate instead of in terms of **po** wrappers.

```
In[27]:= SubstTest[implies,
          and[equal[x, po[y]], member[x, V], equal[domain[GLB[x]], P[fix[x]]]],
          equal[domain[LUB[x]], P[fix[x]]], y → x]

Out[27]= or[equal[domain[LUB[x]], P[fix[x]]], not[equal[domain[GLB[x]], P[fix[x]]]],
           not[member[x, V]], not[PARTIALORDER[x]]] == True

In[28]:= or[equal[domain[LUB[x_]], P[fix[x_]]], not[equal[domain[GLB[x_]], P[fix[x_]]]],
           not[member[x_, V]], not[PARTIALORDER[x_]]] := True

In[29]:= SubstTest[implies,
          and[equal[x, po[y]], member[x, V], equal[domain[LUB[x]], P[fix[x]]]],
          equal[domain[GLB[x]], P[fix[x]]], y → x]

Out[29]= or[equal[domain[GLB[x]], P[fix[x]]], not[equal[domain[LUB[x]], P[fix[x]]]],
           not[member[x, V]], not[PARTIALORDER[x]]] == True

In[30]:= or[equal[domain[GLB[x_]], P[fix[x_]]], not[equal[domain[LUB[x_]], P[fix[x_]]]],
           not[member[x_, V]], not[PARTIALORDER[x_]]] := True
```

Restatement:

```
In[31]:= implies[member[x, PO], equiv[equal[domain[GLB[x]], P[fix[x]]],
             equal[domain[LUB[x]], P[fix[x]]]]] // not // not

Out[31]= True
```