characterization of a complete lattice

Johan G. F. Belinfante
2006 January 4

```
In[1]:= SetDirectory["l:"]; << goedel77.04b; << tools.m

        :Package Title: goedel77.04b       2006 January 4 at 8:20 a.m.

        It is now:  2006 Jan 4 at 9:1

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 January 2

        weightlimit = 40
```

summary

In this notebook a characterization of a complete lattice is derived which has the virtue that it does not make any explicit mention of the concept of greatest lower bound or least upper bound, but instead uses the condition that upper bound sets are vertical sections.

derivation

Lemma. If **y** has an upper bound with respect to a partial order **po[x]**, then **y** must be a subclass of **fix[po[x]]**.

```
In[2]:= SubstTest[implies, not[subclass[y, domain[z]]], equal[ub[z, y], 0], z → po[x]]

Out[2]= or[equal[0, ub[po[x], y]], subclass[y, fix[po[x]]]] == True
```

```
In[3]:= or[equal[0, ub[po[x_], y_]], subclass[y_, fix[po[x_]]]] := True
```

Lemma. If **ub[po[x], y]** is a vertical section of **po[x]** at **z**, then **z** belongs to **fix[po[x]]**.

```
In[4]:= SubstTest[implies, and[member[z, u], equal[u, v]],
          member[z, v], {u → image[po[x], set[z]], v → ub[po[x], y]}] // MapNotNot

Out[4]= or[not[equal[image[po[x], set[z]], ub[po[x], y]]],
          not[member[pair[z, z], po[x]]], subclass[y, image[inverse[po[x]], set[z]]]] == True
```

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

Restatement of the lemma.

```
In[6]:= Map[not, SubstTest[and, implies[p2, p3],
          implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
          {p1 -> equal[ub[po[x], y], image[po[x], set[z]]], p2 -> member[z, fix[po[x]]],
           p3 -> member[pair[z, z], po[x]], p4 -> subclass[y, image[inverse[po[x]], set[z]]]}]]

Out[6]= or[not[equal[image[po[x], set[z]], ub[po[x], y]]],
          not[member[z, fix[po[x]]]], subclass[y, image[inverse[po[x]], set[z]]]] == True

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

If **ub[po[x], y]** is a nonempty vertical section of **po[x]** for some set **y**, then **y** belongs to **domain[LUB[po[x]]]**.

```
In[8]:= Map[not,
         SubstTest[and, implies[and[p2, p3], p4], implies[and[p2, p4], p5], implies[p2, p6],
          implies[and[p1, p2, p4, p5, p6], p7], not[implies[and[p1, p2, p3], p7]],
          {p1 → member[y, V], p2 → equal[ub[po[x], y], image[po[x], set[z]]],
           p3 → not[equal[0, ub[po[x], y]]], p4 → member[z, fix[po[x]]],
           p5 → member[z, ub[po[x], y]],
           p6 -> subclass[intersection[fix[po[x]], ub[po[x], y]], image[po[x], set[z]]],
           p7 -> member[y, domain[LUB[po[x]]]]}]]

Out[8]= or[equal[0, ub[po[x], y]], member[y, domain[LUB[po[x]]]],
          not[equal[image[po[x], set[z]], ub[po[x], y]]], not[member[y, V]]] == True

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

Eliminating the variable **z** is slightly tricky. The following works, albeit it takes quite a while.

```
In[10]:= Map[implies[member[y, z], equal[#, V]] &, SubstTest[class, t,
           or[equal[0, u], member[y, v], not[member[u, image[w, set[t]]]], not[member[y, V]]],
           {u -> ub[po[x], y], v -> domain[LUB[po[x]]], w → VERTSECT[po[x]]}]] // Reverse

Out[10]= or[equal[0, ub[po[x], y]], member[y, domain[LUB[po[x]]]],
           not[member[y, z]], not[member[ub[po[x], y], range[VERTSECT[po[x]]]]]] == True

In[11]:= or[equal[0, ub[po[x_], y_]], member[y_, domain[LUB[po[x_]]]], not[member[y_, z_]],
           not[member[ub[po[x_], y_], range[VERTSECT[po[x_]]]]]] := True
```

---

eliminating the variable y

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

```
In[12]:= Map[equal[V, #] &, SubstTest[class, y, not[member[y, z]],
           z -> intersection[domain[UB[po[x]]], image[inverse[VERTSECT[UB[po[x]]]],
              range[VERTSECT[po[x]]]], complement[domain[LUB[po[x]]]]]]] // Reverse

Out[12]= subclass[intersection[domain[UB[po[x]]], image[
             inverse[VERTSECT[UB[po[x]]]], range[VERTSECT[po[x]]]]], domain[LUB[po[x]]]] == True

In[13]:= (% /. x → x_) /. Equal → SetDelayed
```

This result can be cleaned up. A lemma is needed:

```
In[14]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> dif[P[fix[po[x]]], set[0]],
           v → intersection[domain[UB[po[x]]], image[inverse[VERTSECT[UB[po[x]]]],
              range[VERTSECT[po[x]]]]], w -> domain[LUB[po[x]]]}]

Out[14]= or[not[subclass[P[fix[po[x]]], domain[UB[po[x]]]]], not[subclass[P[fix[po[x]]],
           union[image[inverse[VERTSECT[UB[po[x]]]], range[VERTSECT[po[x]]]], set[0]]]],
          subclass[P[fix[po[x]]], union[domain[LUB[po[x]]], set[0]]]] == True

In[15]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. If a poset has a top element, and if every upper bound set is a vertical section, then every nonempty subset has a lub.

```
In[16]:= Map[implies[member[po[x], y], not[#]] &,
          SubstTest[and, implies[p1, p4], implies[p4, p5], implies[and[p3, p5], p6],
           implies[p2, p7], implies[and[p6, p7], p8], not[implies[and[p1, p2, p3], p8]],
           {p1 → member[po[x], V], p2 → not[empty[ub[po[x], fix[po[x]]]]],
            p3 → equal[range[VERTSECT[po[x]]], range[VERTSECT[UB[po[x]]]]],
            p4 → equal[domain[VERTSECT[UB[po[x]]]], complement[set[0]]],
            p5 -> subclass[dif[P[fix[po[x]]], set[0]], domain[VERTSECT[UB[po[x]]]]],
            p6 -> subclass[dif[P[fix[po[x]]], set[0]],
              image[inverse[VERTSECT[UB[po[x]]]], range[VERTSECT[po[x]]]]],
            p7 -> subclass[dif[P[fix[po[x]]], set[0]], domain[UB[po[x]]]],
            p8 -> subclass[dif[P[fix[po[x]]], set[0]], domain[LUB[po[x]]]]}]]

Out[16]= or[equal[0, ub[po[x], fix[po[x]]]], not[
           equal[range[VERTSECT[po[x]]], range[VERTSECT[UB[po[x]]]]]], not[member[po[x], y]],
          subclass[P[fix[po[x]]], union[domain[LUB[po[x]]], set[0]]]] == True

In[17]:= or[equal[0, ub[po[x_], fix[po[x_]]]],
           not[equal[range[VERTSECT[po[x_]]], range[VERTSECT[UB[po[x_]]]]]],
           not[member[po[x_], y_]],
           subclass[P[fix[po[x_]]], union[domain[LUB[po[x_]]], set[0]]]] := True
```

a characterization of a complete lattice

If a poset has a top and bottom element, and if every upper bound set is a vertical section, then the poset is a complete lattice.

```
In[18]:= Map[implies[member[po[x], y], not[#]] &, SubstTest[and, implies[and[p1, p4, p5], p6],
           implies[and[p2, p3], p7], implies[and[p6, p7], p8], implies[and[p1, p8], p9],
           not[implies[and[p1, p2, p3, p4, p5], p9]], {p1 → member[po[x], V],
            p2 → not[equal[0, po[x]]], p3 → not[equal[0, lb[po[x], fix[po[x]]]]],
            p4 → not[equal[0, ub[po[x], fix[po[x]]]]],
            p5 → equal[range[VERTSECT[po[x]]], range[VERTSECT[UB[po[x]]]]],
            p6 -> subclass[P[fix[po[x]]], union[domain[LUB[po[x]]], set[0]]],
            p7 → member[0, domain[LUB[po[x]]]],
            p8 -> subclass[P[fix[po[x]]], domain[LUB[po[x]]], p9 → member[po[x], CL]}]]

Out[18]= or[equal[0, lb[po[x], fix[po[x]]]], equal[0, po[x]], equal[0, ub[po[x], fix[po[x]]]],
           member[po[x], CL], not[equal[range[VERTSECT[po[x]]], range[VERTSECT[UB[po[x]]]]],
           not[member[po[x], y]]] == True

In[19]:= or[equal[0, lb[po[x_], fix[po[x_]]]], equal[0, po[x_]],
           equal[0, ub[po[x_], fix[po[x_]]]], member[po[x_], CL],
           not[equal[range[VERTSECT[po[x_]]], range[VERTSECT[UB[po[x_]]]]]],
           not[member[po[x_], y_]]] := True
```

Unwrapped version:

```
In[20]:= SubstTest[implies,
           and[equal[x, po[z]], member[x, y], not[equal[0, x]], not[equal[0, lb[x, fix[x]]]],
            not[equal[0, ub[x, fix[x]]]], equal[range[VERTSECT[x]], range[VERTSECT[UB[x]]]],
           member[x, CL], z → x]

Out[20]= or[equal[0, x], equal[0, lb[x, fix[x]]], equal[0, ub[x, fix[x]]],
           member[x, CL], not[equal[range[VERTSECT[x]], range[VERTSECT[UB[x]]]],
           not[member[x, y]], not[PARTIALORDER[x]]] == True

In[21]:= or[equal[0, lb[x_, fix[x_]]], equal[0, x_], equal[0, ub[x_, fix[x_]]],
           member[x_, CL], not[equal[range[VERTSECT[x_]], range[VERTSECT[UB[x_]]]],
           not[member[x_, y_]], not[PARTIALORDER[x_]]] := True
```