cross products of complete lattices

Johan G. F. Belinfante
2006 January 4

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

        :Package Title: goedel77.04c       2006 January 4 at 9:30 a.m.

        It is now:  2006 Jan 4 at 11:21

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 January 2

        weightlimit = 40
```

summary

The cross product of complete lattices is a complete lattice. The derivation of this fact in this notebook makes use of a characterization of a complete lattice as a poset with top and bottom elements with the property that every set of upper bounds is a final segment, that is, a vertical section of the partial order relation.

a characterization of complete lattices

For cross-products, the characterization of complete lattices reduces to this:

```
In[2]:= SubstTest[implies, and[not[equal[0, z]],
          member[z, PO], not[empty[lb[z, fix[z]]]], not[empty[ub[z, fix[z]]]],
          equal[range[VERTSECT[z]], range[VERTSECT[UB[z]]]], member[z, CL], z → cross[x, y]]

Out[2]= or[equal[0, domain[x]], equal[0, domain[y]],
          equal[0, lb[cross[x, y], cart[fix[x], fix[y]]]],
          equal[0, ub[cross[x, y], cart[fix[x], fix[y]]]], member[cross[x, y], CL],
          not[equal[range[VERTSECT[cross[x, y]]], union[image[VERTSECT[cross[UB[x], UB[y]]],
              cart[complement[set[0]], complement[set[0]]], set[0]]]],
          not[member[cross[x, y], V]], not[PARTIALORDER[cross[x, y]]] == True

In[3]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

In the remainder of this notebook, each of these conditions will be shown to be true when $\mathbf{x}$ and $\mathbf{y}$ are complete lattices.

the domain condition

The validity of the two domain conditions in the preceding section is derived as follows:

```
In[4]:= Map[not, SubstTest[and, implies[p1, p2],
          implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
          {p1 → member[x, CL], p2 → not[empty[fix[x]]],
           p3 → equal[fix[x], domain[x]], p4 → not[empty[domain[x]]]}]]

Out[4]= or[not[equal[0, domain[x]]], not[member[x, CL]]] == True

In[5]:= or[not[equal[0, domain[x_]]], not[member[x_, CL]]] := True
```

top and bottom

Lemma 1.

```
In[6]:= SubstTest[implies,
         and[not[empty[t]], not[empty[w]], subclass[cart[t, w], z]], not[empty[z]],
         {t → lb[x, u], w → lb[y, v], z → lb[cross[x, y], cart[u, v]]}]

Out[6]= or[equal[0, lb[x, u]], equal[0, lb[y, v]],
          not[equal[0, lb[cross[x, y], cart[u, v]]]]] == True

In[7]:= or[equal[0, lb[x_, u_]], equal[0, lb[y_, v_]],
          not[equal[0, lb[cross[x_, y_], cart[u_, v_]]]]] := True
```

Lemma 2.

```
In[8]:= SubstTest[implies,
         and[not[empty[t]], not[empty[w]], subclass[cart[t, w], z]], not[empty[z]],
         {t → ub[x, u], w → ub[y, v], z → ub[cross[x, y], cart[u, v]]}]

Out[8]= or[equal[0, ub[x, u]], equal[0, ub[y, v]],
          not[equal[0, ub[cross[x, y], cart[u, v]]]]] == True

In[9]:= or[equal[0, ub[x_, u_]], equal[0, ub[y_, v_]],
          not[equal[0, ub[cross[x_, y_], cart[u_, v_]]]]] := True
```

Since complete lattices **x** and **y** have top and bottom elements, so does **cross[x,y]**.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4], implies[and[p3, p4], p5],
           not[implies[and[p1, p2], p5]], {p1 → member[x, CL], p2 → member[y, CL],
            p3 → not[empty[ub[x, fix[x]]]], p4 → not[empty[ub[y, fix[y]]]],
            p5 → not[empty[ub[cross[x, y], cart[fix[x], fix[y]]]]]}]]

Out[10]= or[not[equal[0, ub[cross[x, y], cart[fix[x], fix[y]]]]],
           not[member[x, CL]], not[member[y, CL]]] == True
```

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

```
In[12]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4], implies[and[p3, p4], p5],
           not[implies[and[p1, p2], p5]], {p1 → member[x, CL], p2 → member[y, CL],
            p3 → not[empty[lb[x, fix[x]]]], p4 → not[empty[lb[y, fix[y]]]],
            p5 → not[empty[lb[cross[x, y], cart[fix[x], fix[y]]]]]}]]
```

```
Out[12]= or[not[equal[0, lb[cross[x, y], cart[fix[x], fix[y]]]]],
           not[member[x, CL]], not[member[y, CL]]] == True
```

```
In[13]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

PARTIALORDER

The cross product of complete lattices is a partial order.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4], implies[and[p3, p4], p5],
           not[implies[and[p1, p2], p5]], {p1 → member[x, CL], p2 -> member[y, CL],
            p3 → PARTIALORDER[x], p4 → PARTIALORDER[y], p5 -> PARTIALORDER[cross[x, y]]}]]
```

```
Out[14]= or[not[member[x, CL]], not[member[y, CL]], PARTIALORDER[cross[x, y]]] == True
```

```
In[15]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

---

vertical section condition

The following temporary abbreviation is introduced for the vertical section condition:

```
In[16]:= ravs[x_] := equal[range[VERTSECT[x]], range[VERTSECT[UB[x]]]]
```

Lemma. The vertical section condition is preserved when **x** and **y** are sets.

```
In[17]:= Map[implies[and[member[x, CL], member[y, CL]], #] &,
          SubstTest[implies, and[equal[x, setpart[u]], equal[y, setpart[v]], ravs[x], ravs[y]],
           ravs[cross[x, y]], {u → x, v → y}]]
```

```
Out[17]= or[equal[range[VERTSECT[cross[x, y]]], union[image[VERTSECT[cross[UB[x], UB[y]]],
              cart[complement[set[0]], complement[set[0]]], set[0]]],
           not[equal[range[VERTSECT[x]], range[VERTSECT[UB[x]]]]],
           not[equal[range[VERTSECT[y]], range[VERTSECT[UB[y]]]]],
           not[member[x, CL]], not[member[y, CL]]] == True
```

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

This can be cleaned up.

```
In[19]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
           implies[and[p1, p2, p3, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 -> member[x, CL], p2 -> member[y, CL], p3 → ravs[x],
            p4 → ravs[y], p5 → ravs[cross[x, y]]}]]
```

```
Out[19]= or[equal[range[VERTSECT[cross[x, y]]], union[image[VERTSECT[cross[UB[x], UB[y]]],
             cart[complement[set[0]], complement[set[0]]]], set[0]]],
           not[member[x, CL]], not[member[y, CL]]] == True
```

```
In[20]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

---

put it all together

The main theorem is established by putting together all the lemmas derived in the preceding sections.

```
In[21]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p1, p4],
           implies[p1, p5], implies[p1, p6], implies[p1, p7], implies[p1, p8],
           implies[and[p2, p3, p4, p5, p6, p7, p8], p9], not[implies[p1, p9]],
           {p1 → and[member[x, CL], member[y, CL]], p2 → ravs[cross[x, y]],
            p3 → member[cross[x, y], V], p4 → not[equal[0, domain[x]]], p5 →
             not[equal[0, domain[y]]], p6 → not[equal[0, lb[cross[x, y], cart[fix[x], fix[y]]]]],
            p7 → not[equal[0, ub[cross[x, y], cart[fix[x], fix[y]]]]],
            p8 → PARTIALORDER[cross[x, y]], p9 → member[cross[x, y], CL]}]]
```

```
Out[21]= or[member[cross[x, y], CL], not[member[x, CL]], not[member[y, CL]]] == True
```

```
In[22]:= or[member[cross[x_, y_], CL], not[member[x_, CL]], not[member[y_, CL]]] := True
```

A variable-free statement is possible:

```
In[25]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[x, y], member[pair[x, y], z],
            z → union[complement[cart[CL, CL]], image[inverse[CROSS], CL]]]] // Reverse
```

```
Out[25]= subclass[image[CROSS, cart[CL, CL]], CL] == True
```

```
In[26]:= subclass[image[CROSS, cart[CL, CL]], CL] := True
```