

CL = the set of all complete lattices

Johan G. F. Belinfante
2005 October 6

```
In[1]:= SetDirectory["1:"]; << goedel74.06a; << tools.m

:Package Title: goedel74.06a          2005 October 6 at 4:00 p.m.

It is now: 2005 Oct 6 at 22:32

Loading Simplification Rules

TOOLS.M          Revised 2005 October 2

weightlimit = 40
```

summary

This notebook introduces the set **CL** of partial order relations for complete lattices. Instead of regarding a complete lattice as a set equipped with glb and lub operations, the focus here is on the underlying partial order. Thus the class **CL** is regarded as a certain subclass of the class **PO** of all partial order relations. To avoid having statements about membership in **CL** expanded out, the definition of **CL** is given in the form of a rewrite rule for **image[V,intersection[CL, set[x]]]**. A few basic properties and examples of complete lattices are considered here.

definition of a complete lattice

The definition of a complete lattice is presented here as the following rewrite rule.

```
In[2]:= image[V, intersection[CL, set[x_]]] := intersection[
    complement[image[V, intersection[complement[domain[GLB[x]], P[fix[x]]]]],
    image[V, intersection[PO, set[x]]]]
```

From this, a membership rule is derived, but it is turned around so that it does not expand out.

```
In[3]:= member[x, CL] // AssertTest // Reverse

Out[3]= and[member[x, V], PARTIALORDER[x], subclass[P[fix[x]], domain[GLB[x]]] == member[x, CL]

In[4]:= and[member[x_, V], PARTIALORDER[x_],
    subclass[P[fix[x_]], domain[GLB[x_]]] := member[x, CL]
```

some examples

The empty set relation is not a complete lattice.

```
In[5]:= member[0, CL] // AssertTest
```

```
Out[5]= member[0, CL] == False
```

```
In[6]:= member[0, CL] := False
```

The identity relation on any singleton is a complete lattice.

```
In[7]:= member[cart[set[x], set[x]], CL] // AssertTest
```

```
Out[7]= member[cart[set[x], set[x]], CL] == member[x, V]
```

```
In[8]:= member[cart[set[x_], set[x_]], CL] := member[x, V]
```

The divisibility relation for natural numbers provides a simple example:

```
In[9]:= member[DIV, CL] // AssertTest
```

```
Out[9]= member[DIV, CL] == True
```

```
In[10]:= member[DIV, CL] := True
```

Another important example is the subset relation restricted to a power set. A lemma is needed.

```
In[11]:= SubstTest[PARTIALORDER, restrict[po[s], t, t], {s → S, t → P[x]}]
```

```
Out[11]= PARTIALORDER[composite[id[P[x]], S]] == True
```

```
In[12]:= PARTIALORDER[composite[id[P[x_]], S]] := True
```

Theorem.

```
In[13]:= member[composite[id[P[x]], S], CL] // AssertTest
```

```
Out[13]= member[composite[id[P[x]], S], CL] == member[x, V]
```

```
In[14]:= member[composite[id[P[x_]], S], CL] := member[x, V]
```

CL as a subclass of PO

Lemma.

```
In[15]:= (implies[member[x, CL], member[x, y]] // AssertTest) /. y → PO
```

```
Out[15]= or[not[member[x, CL]], PARTIALORDER[x]] == True
```

```
In[16]:= or[not[member[x_, CL]], PARTIALORDER[x_]] := True
```

```
In[17]:= subclass[CL, PO] // AssertTest
```

```
Out[17]= subclass[CL, PO] == True
```

```
In[18]:= subclass[CL, PO] := True
```

The following facts are immediate corollaries of the definition.

```
In[19]:= SubstTest[implies, and[p1, p2], p2,
  {p1 → member[x, PO], p2 → subclass[P[fix[x]], domain[GLB[x]]]}]
```

```
Out[19]= or[not[member[x, CL]], subclass[P[fix[x]], domain[GLB[x]]]] == True
```

```
In[20]:= or[not[member[x_, CL]], subclass[P[fix[x_]], domain[GLB[x_]]]] := True
```

Lemma.

```
In[21]:= Map[or[#, equal[y, P[x]]] &, not[equal[P[x], y]] // AssertTest // Reverse]
```

```
Out[21]= or[equal[y, P[x]], not[subclass[P[x], y]], not[subclass[U[y], x]]] == True
```

```
In[22]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. (Replacing the inclusion for $\text{domain}[\text{GLB}[x]]$ with an equation.)

```
In[23]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p2, p4],
  implies[and[p2, p3, p4], p5], implies[and[p3, p5], p6], not[implies[p1, p6]],
  {p1 → member[x, CL], p2 → member[x, PO], p3 → subclass[P[fix[x]], domain[GLB[x]]],
  p4 → equal[fix[x], range[x]], p5 → subclass[U[domain[GLB[x]]], fix[x]],
  p6 → equal[domain[GLB[x]], P[fix[x]]]}]]]
```

```
Out[23]= or[equal[domain[GLB[x]], P[fix[x]]], not[member[x, CL]]] == True
```

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

Corollary. (One can replace $\text{GLB}[x]$ with $\text{LUB}[x]$.)

```
In[25]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
  not[implies[p1, p4]], {p1 → member[x, CL], p2 → member[x, PO],
  p3 → equal[domain[GLB[x]], P[fix[x]]], p4 → equal[domain[LUB[x]], P[fix[x]]]}]]]
```

```
Out[25]= or[equal[domain[LUB[x]], P[fix[x]]], not[member[x, CL]]] == True
```

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

duality

If \mathbf{x} is a complete lattice, so is $\text{inverse}[\mathbf{x}]$. Some temporary lemmas are needed to derive this.

```
In[27]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
  {p1 → member[x, CL], p2 → PARTIALORDER[x], p3 → PARTIALORDER[composite[Id, x]]}]
```

```
Out[27]= or[not[member[x, CL]], PARTIALORDER[composite[Id, x]]] == True
```

```
In[28]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[29]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
  {p1 → member[x, CL], p2 → equal[domain[LUB[x]], P[fix[x]]],
  p3 → subclass[P[fix[x]], domain[LUB[x]]]}]
```

```
Out[29]= or[not[member[x, CL]], subclass[P[fix[x]], domain[LUB[x]]]] == True
```

```
In[30]:= (% /. x → x_) /. Equal → SetDelayed
```

Duality Theorem.

```
In[31]:= implies[member[x, CL], member[inverse[x], CL]] // AssertTest // MapNotNot
```

```
Out[31]= or[member[inverse[x], CL], not[member[x, CL]]] == True
```

```
In[32]:= or[member[inverse[x_], CL], not[member[x_, CL]]] := True
```

lemma

The rewrite rule derived in this section is used in the next section.

```
In[33]:= SubstTest[implies, and[equal[x, z], equal[x, image[w, z]]], equal[x, image[w, x]],
  {w → IMAGE[id[y]], z → intersection[P[y], x]}]
```

```
Out[33]= or[equal[x, image[IMAGE[id[y]], x]], not[subclass[U[x], y]]] == True
```

```
In[34]:= or[equal[x_, image[IMAGE[id[y_]], x_]], not[subclass[U[x_], y_]]] := True
```

A new rewrite rule can be introduced as follows:

```
In[35]:= equiv[equal[image[IMAGE[id[y]], x], x], subclass[U[x], y]]
```

```
Out[35]= True
```

```
In[36]:= equal[x_, image[IMAGE[id[y_]], x_]] := subclass[U[x], y]
```

variable-free formulation of duality

Variable-free formulation.

```
In[37]:= Map[equal[V, #] &,
            SubstTest[class, x, or[member[inverse[x], y], not[member[x, y]]], y → CL]] // Reverse
```

```
Out[37]= subclass[image[IMAGE[SWAP], CL], CL] == True
```

```
In[38]:= % /. Equal → SetDelayed
```

Lemma.

```
In[39]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
                {u → CL, v → PO, w → P[cart[V, V]]}]
```

```
Out[39]= subclass[U[CL], cart[V, V]] == True
```

```
In[40]:= % /. Equal → SetDelayed
```

The result of the preceding section can now be applied:

```
In[41]:= equal[image[IMAGE[id[cart[V, V]]], CL], CL]
```

```
Out[41]= True
```

```
In[42]:= image[IMAGE[id[cart[V, V]]], CL] := CL
```

The invariance of **CL** under **IMAGE[SWAP]** implies that the reverse inclusion also holds:

```
In[43]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
                {u → image[IMAGE[SWAP], CL], v → CL, w → IMAGE[SWAP]}]
```

```
Out[43]= subclass[CL, image[IMAGE[SWAP], CL]] == True
```

```
In[44]:= % /. Equal → SetDelayed
```

The duality property of **CL** can therefore be expressed as an equation:

```
In[45]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → image[IMAGE[SWAP], CL], v → CL}]
```

```
Out[45]= True == equal[CL, image[IMAGE[SWAP], CL]]
```

```
In[46]:= image[IMAGE[SWAP], CL] := CL
```