greatest and least elements for partial orders

Johan G. F. Belinfante
2010 January 16

```
In[1]:= SetDirectory["l:"]; << goedel.10jan15a;<< tools.m

        :Package Title: goedel.10jan15a              2010 January 15 at 1:30 p.m.

        It is now:  2010 Jan 16 at 20:49

        Loading Simplification Rules

        TOOLS.M                                    Revised 2010 January 7

        weightlimit = 40
```

summary

When **x** is a partial order, a subset **y** of its fixed-point class **fix[x]** may or may not have a least and/or a greatest element, but when it does, such elements are unique. These uniqueness properties are succinctly expressed by the statements that the relations **LEAST[x]** and **GREATEST[x]** are functions when **x** is a partial order. Accordingly, there are technically two different ways of talking about least and greatest elements of subsets **y ⊂ fix[x]**, a distinction which unfortunately tends to get blurred over by common parlance. The first way is to say that least and greatest elements are members of classes **least[x, y]** and **greatest[x, y]**, respectively. The second way of talking about least and greatest elements is to say they are obtained by applying one of the two functions **LEAST[x]** or **GREATEST[x]** to the set **y**. In this notebook rewrite rules are derived connecting these two different descriptions of least and greatest elements.

It should be noted that for some partial order relations (for example, the subset relation **S**), the class **fix[x]** may be a proper class. When this is the case, a proper class **y** contained in **fix[x]** could have a least or greatest element in the sense of belonging to **least[x, y]** or **greatest[x, y]**, but such elements can never be obtained by applying either of the functions **LEAST[x]** or **GREATEST[x]** to **y**. The argument of a function must always be a set, never a proper class. The problem of connecting the two different descriptions of least and greatest elements therefore only arises for the case that **y** is a set. This is not to say that the functions **LEAST[x]** and **GREATEST[x]** are uninteresting when **x** is a proper class. These two functions can still be applied to subsets of **fix[x]** that have a least or greatest element.

Each of the three theorems about greatest elements of partial orders are stated twice, once using the **po** wrapper, and again using the **PARTIALORDER** predicate. Applying duality, an additional six similar rewrite rules are derived for least elements. In the final section of this notebook the case of complete lattices is considered in some detail. For several theorems significant speedups were achieved by omitting proof steps.

For convenience to the reader, most theorems are accompanied by informal descriptions of their contents, but such informal descriptions may be somewhat imprecise, and should never be taken as substitutes for the actual theorems.

derivation

It is convenient to begin by using the **po** wrapper for partial orders, and then later to remove these wrappers in favor of the **PARTIALORDER** predicate. Whenever a sethood hypothesis is needed, instead of making this hypothesis in the form **y ∈ V**, a new class variable **z** is introduced and the sethood hypothesis is formulated in the slightly more general form **y ∈ z**. This is done solely for the sake of making pattern matching more convenient for applications of the rewrite rules that are derived. Note that the abbreviation **greatest[x, y]** automatically expands out to **y ⋂ ub[x, y]**, and similarly for the abbreviation **least[x, y]**.

Theorem. If a set **y** has a greatest element, then applying the function **GREATEST[po[x]]** to the set **y** produces a member of **y**.

```
In[2]:= Map[implies[member[y, z], or[member[APPLY[GREATEST[po[x]], y], y], #]] &, Map[
            implies[member[y, domain[GREATEST[po[x]]]], member[APPLY[GREATEST[po[x]], y], #]] &,
            SubstTest[image, funpart[t], set[y], t → GREATEST[po[x]]]] // Reverse // MapNotNot

Out[2]= or[equal[0, intersection[y, ub[po[x], y]]],
          member[APPLY[GREATEST[po[x]], y], y], not[member[y, z]]] == True

In[3]:= or[equal[0, intersection[y_, ub[po[x_], y_]]],
          member[APPLY[GREATEST[po[x_]], y_], y_], not[member[y_, z_]]] := True
```

The dual statement follows as an immediate corollary.

Corollary. If a set **y** has a least element, then applying the function **LEAST[po[x]]** to **y** produces a member of **y**.

```
In[4]:= SubstTest[or, equal[0, intersection[y, ub[po[t], y]]],
          member[APPLY[GREATEST[po[t]], y], y], not[member[y, z]], t → inverse[po[x]]] // Reverse

Out[4]= or[equal[0, intersection[y, lb[po[x], y]]],
          member[APPLY[LEAST[po[x]], y], y], not[member[y, z]]] == True

In[5]:= or[equal[0, intersection[y_, lb[po[x_], y]]],
          member[APPLY[LEAST[po[x_]], y_], y_], not[member[y_, z_]]] := True
```

The **po** wrapper can be eliminated in favor of a **PARTIALORDER** predicate in the standard fashion.

Theorem. If **x** is a partial order, and **y** is a set with a greatest element, the applying **GREATEST[x]** to **y** yields a member of **y**.

```
In[6]:= SubstTest[implies, equal[x, po[t]], or[equal[0, intersection[y, ub[x, y]]],
            member[APPLY[GREATEST[x], y], y], not[member[y, z]]], t → x] // Reverse

Out[6]= or[equal[0, intersection[y, ub[x, y]]],
          member[APPLY[GREATEST[x], y], y], not[member[y, z]], not[PARTIALORDER[x]]] == True

In[7]:= or[equal[0, intersection[y_, ub[x_, y_]]], member[APPLY[GREATEST[x_], y_], y_],
          not[member[y_, z_]], not[PARTIALORDER[x_]]] := True
```

The dual result is obtained in a similar fashion.

Corollary. If **x** is a partial order, and if **y** is a set with a least element, the applying **LEAST[x]** to **y** yields a member of **y**.

```
In[8]:= SubstTest[implies, equal[x, po[t]], or[equal[0, intersection[y, lb[x, y]]],
          member[APPLY[LEAST[x], y], y], not[member[y, z]]], t → x] // Reverse

Out[8]= or[equal[0, intersection[y, lb[x, y]]],
          member[APPLY[LEAST[x], y], y], not[member[y, z]], not[PARTIALORDER[x]]] == True

In[9]:= or[equal[0, intersection[y_, lb[x_, y_]]], member[APPLY[LEAST[x_], y_], y_],
          not[member[y_, z_]], not[PARTIALORDER[x_]]] := True
```

bound properties

Theorem. If a set **y** has a greatest element, then applying the function **GREATEST[po[x]]** to **y** produces an upper bound for **y**.

```
In[10]:= Map[implies[and[member[y, z], member[y, domain[GREATEST[po[x]]]]],
            member[APPLY[GREATEST[po[x]], y], #]] &,
          SubstTest[image, funpart[t], set[y], t → GREATEST[po[x]]] // Reverse] // MapNotNot

Out[10]= or[equal[0, intersection[y, ub[po[x], y]]], not[member[y, z]],
          subclass[y, image[inverse[po[x]], set[APPLY[GREATEST[po[x]], y]]]] == True

In[11]:= or[equal[0, intersection[y_, ub[po[x_], y_]]], not[member[y_, z_]],
          subclass[y_, image[inverse[po[x_]], set[APPLY[GREATEST[po[x_]], y_]]]]] := True
```

The preceding theorem and the first theorem of the preceding section can be combined into the following statement. (Comment: The use of double negation suffices to allow both rewrite rules to simplify the combined statement to **True**.)

```
In[12]:= implies[and[member[y, z], not[empty[greatest[po[x], y]]]],
            member[APPLY[GREATEST[po[x]], y], greatest[po[x], y]]] // not // not

Out[12]= True
```

The dual result is obtained by simply replacing **x** with **inverse[po[x]]**.

Corollary. If a set **y** has a least element, then applying the function **LEAST[po[x]]** to **y** produces a lower bound for **y**.

```
In[13]:= SubstTest[or, equal[0, intersection[y, ub[po[t], y]]], not[member[y, z]],
          subclass[y, image[inverse[po[t]], set[APPLY[GREATEST[po[t]], y]]]],
          t → inverse[po[x]]] // Reverse

Out[13]= or[equal[0, intersection[y, lb[po[x], y]]], not[member[y, z]],
          subclass[y, image[po[x], set[APPLY[LEAST[po[x]], y]]]] == True

In[14]:= or[equal[0, intersection[y_, lb[po[x_], y_]]], not[member[y_, z_]],
          subclass[y_, image[po[x_], set[APPLY[LEAST[po[x_]], y_]]]]] := True
```

Restatement:

```
In[15]:= implies[and[member[y, z], not[empty[least[po[x], y]]]],
           member[APPLY[LEAST[po[x]], y], least[po[x], y]]] // not // not

Out[15]= True
```

Both theorems in this section can be restated with the wrapper **po** replaced by a **PARTIALORDER** predicate.

Corollary. If **x** is a partial order and **y** is a set with a greatest element, then **APPLY[GREATEST[x], y]** is an upper bound for **y**.

```
In[16]:= SubstTest[implies, and[equal[x, po[t]], member[y, z], not[empty[greatest[x, y]]]],
           member[APPLY[GREATEST[x], y], ub[x, y]], t → x] // Reverse

Out[16]= or[equal[0, intersection[y, ub[x, y]]], not[member[y, z]], not[PARTIALORDER[x]],
           subclass[y, image[inverse[x], set[APPLY[GREATEST[x], y]]]]] == True

In[17]:= or[equal[0, intersection[y_, ub[x_, y_]]], not[member[y_, z_]], not[PARTIALORDER[x_]],
           subclass[y_, image[inverse[x_], set[APPLY[GREATEST[x_], y_]]]]] := True
```

Dual result:

Corollary. If **x** is a partial order and **y** is a set with a least element, then **APPLY[LEAST[x], y]** is a lower bound for **y**.

```
In[18]:= SubstTest[implies, and[equal[x, po[t]], member[y, z], not[empty[least[x, y]]]],
           member[APPLY[LEAST[x], y], lb[x, y]], t → x] // Reverse

Out[18]= or[equal[0, intersection[y, lb[x, y]]], not[member[y, z]],
           not[PARTIALORDER[x]], subclass[y, image[x, set[APPLY[LEAST[x], y]]]]] == True

In[19]:= or[equal[0, intersection[y_, lb[x_, y_]]], not[member[y_, z_]],
           not[PARTIALORDER[x_]], subclass[y_, image[x_, set[APPLY[LEAST[x_], y_]]]]] := True
```

---

uniqueness

Theorem. If **w** is a greatest element of a set **y**, then **w = APPLY[GREATEST[po[x], y]**.

```
In[20]:= Map[implies[and[member[y, z], member[w, #]], equal[w, APPLY[GREATEST[po[x]], y]]] &,
           SubstTest[image, funpart[t], set[y], t → GREATEST[po[x]]]] // MapNotNot // Reverse

Out[20]= or[equal[w, APPLY[GREATEST[po[x]], y]], not[member[w, y]],
           not[member[y, z]], not[subclass[y, image[inverse[po[x]], set[w]]]]] == True

In[21]:= or[equal[w_, APPLY[GREATEST[po[x_]], y_]], not[member[w_, y_]],
           not[member[y_, z_]], not[subclass[y_, image[inverse[po[x_]], set[w_]]]]] := True
```

The dual result is obtained by replacing **x** with **inverse[po[x]]**.

Theorem. If **w** is a least element of a set **y**, then **w = APPLY[LEAST[po[x], y]**.

```
In[22]:= SubstTest[or, equal[w, APPLY[GREATEST[po[t]], y]], not[member[w, y]], not[member[y, z]],
           not[subclass[y, image[inverse[po[t]], set[w]]]], t → inverse[po[x]]] // Reverse

Out[22]= or[equal[w, APPLY[LEAST[po[x]], y]], not[member[w, y]],
           not[member[y, z]], not[subclass[y, image[po[x], set[w]]]]] == True

In[23]:= or[equal[w_, APPLY[LEAST[po[x_]], y_]], not[member[w_, y_]],
           not[member[y_, z_]], not[subclass[y_, image[po[x_], set[w_]]]]] := True
```

The **po** wrapper can be replaced with **PARTIALORDER**.

Theorem. If **x** is a partial order, and if **w ∈ greatest[x, y]** for a set **y**, then **w = APPLY[GREATEST[x], y]**.

```
In[24]:= SubstTest[implies, equal[x, po[t]],
           or[equal[w, APPLY[GREATEST[x], y]], not[member[w, y]], not[member[y, z]],
            not[subclass[y, image[inverse[x], set[w]]]]], t → x] // Reverse

Out[24]= or[equal[w, APPLY[GREATEST[x], y]], not[member[w, y]], not[member[y, z]],
           not[PARTIALORDER[x]], not[subclass[y, image[inverse[x], set[w]]]]] == True

In[25]:= or[equal[w_, APPLY[GREATEST[x_], y_]], not[member[w_, y_]], not[member[y_, z_]],
           not[PARTIALORDER[x_]], not[subclass[y_, image[inverse[x_], set[w_]]]]] := True
```

Dual result.

Corollary. If **x** is a partial order, and if **w ∈ least[x, y]** for a set **y**, then **w = APPLY[LEAST[x], y]**.

```
In[26]:= SubstTest[implies, equal[x, po[t]], or[equal[w, APPLY[LEAST[x], y]], not[member[w, y]],
            not[member[y, z]], not[subclass[y, image[x, set[w]]]]], t → x] // Reverse

Out[26]= or[equal[w, APPLY[LEAST[x], y]], not[member[w, y]], not[member[y, z]],
           not[PARTIALORDER[x]], not[subclass[y, image[x, set[w]]]]] == True

In[27]:= or[equal[w_, APPLY[LEAST[x_], y_]], not[member[w_, y_]], not[member[y_, z_]],
           not[PARTIALORDER[x_]], not[subclass[y_, image[x_, set[w_]]]]] := True
```

application to complete lattices

A complete lattice order **x ∈ CL** is a (small) partial order for which every subset of **fix[x]** has a **lub** and a **glb**. The set **fix[x]** itself always has both a least and a greatest element, but in general, a subset **y ⊂ fix[x]** need not have either. The following basic theorem allows one to dispense with explicit sethood hypotheses for **y**.

Theorem. If **x ∈ CL** and **y ⊂ fix[x]**, then **y** is a set.

```
In[28]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[and[p1, p2], p4]], {p1 -> member[x, CL], p2 -> subclass[y, fix[x]],
             p3 → member[fix[x], V], p4 -> member[y, V]}]] // Reverse

Out[28]= or[member[y, V], not[member[x, CL]], not[subclass[y, fix[x]]]] == True
```

```
In[29]:= or[member[y_, V], not[member[x_, CL]], not[subclass[y_, fix[x_]]]] := True
```

The derivation in the following theorem is somewhat unusual in that it relies on an **image[V, –]** rule for complete lattices.

Theorem. If **x** is a complete lattice, and if **y** has a greatest element, then **y** is a subset of **fix[x]**.

```
In[30]:= SubstTest[equal, V,
          union[complement[y], complement[image[V, intersection[y, complement[fix[x]]]]],
           complement[image[V, intersection[z, set[x]]]], complement[ub[x, y]]], z → CL]

Out[30]= or[equal[0, intersection[y, ub[x, y]]], not[member[x, CL]], subclass[y, fix[x]]] == True

In[31]:= or[equal[0, intersection[y_, ub[x_, y_]]],
           not[member[x_, CL]], subclass[y_, fix[x_]]] := True
```

Dual result.

Corollary. If **x** is a complete lattice, and if **y** has a least element, then **y** is a subset of **fix[x]**.

```
In[32]:= Map[implies[member[x, CL], #] &, SubstTest[or, equal[0, intersection[y, ub[t, y]]],
            not[member[t, CL]], subclass[y, fix[t]], t → inverse[x]]] // Reverse

Out[32]= or[equal[0, intersection[y, lb[x, y]]], not[member[x, CL]], subclass[y, fix[x]]] == True

In[33]:= or[equal[0, intersection[y_, lb[x_, y_]]],
           not[member[x_, CL]], subclass[y_, fix[x_]]] := True
```

A speedup by a factor 9 was achieved in the following theorem by omitting these three proof steps:

implies[p1, p3], **implies[and[p1, p2], p4]**, **implies[and[p1, p4], p5]**.

Theorem. If **x** is a complete lattice and if **y** has a least element, then **APPLY[LEAST[x], y] ∈ y**.

```
In[34]:= Map[not, SubstTest[and, implies[and[p2, p3, p4, p5], p6],
            not[implies[and[p1, p2], p6]], {p1 → member[x, CL],
             p2 -> not[empty[least[x, y]]], p3 → PARTIALORDER[x], p4 → subclass[y, fix[x]],
             p5 → member[y, V], p6 -> member[APPLY[LEAST[x], y], y]}]] // Reverse

Out[34]= or[equal[0, intersection[y, lb[x, y]]],
           member[APPLY[LEAST[x], y], y], not[member[x, CL]]] == True

In[35]:= or[equal[0, intersection[y_, lb[x_, y_]]],
           member[APPLY[LEAST[x_], y_], y_], not[member[x_, CL]]] := True
```

Dual result.

Corollary. If **x** is a complete lattice and if **y** has a greatest element, then **APPLY[GREATEST[x], y] ∈ y**.

```
In[36]:= Map[implies[member[x, CL], #] &, SubstTest[or, equal[0, intersection[y, lb[t, y]]],
           member[APPLY[LEAST[t], y], y], not[member[t, CL]], t → inverse[x]]] // Reverse

Out[36]= or[equal[0, intersection[y, ub[x, y]]],
          member[APPLY[GREATEST[x], y], y], not[member[x, CL]]] == True

In[37]:= or[equal[0, intersection[y_, ub[x_, y_]]],
          member[APPLY[GREATEST[x_], y_], y_], not[member[x_, CL]]] := True
```

In the derivation of the bounds theorem for complete lattices the same proof steps are omitted as before.

Theorem. If **x** is a complete lattice and if **y** has a least element, then **APPLY[LEAST[x], y]** is a lower bound for **y**.

```
In[38]:= Map[not, SubstTest[and, implies[and[p2, p3, p4, p5], p6],
           not[implies[and[p1, p2], p6]], {p1 → member[x, CL], p2 -> not[empty[least[x, y]]],
            p3 → PARTIALORDER[x], p4 → subclass[y, fix[x]], p5 → member[y, V],
            p6 -> subclass[y, image[x, set[APPLY[LEAST[x], y]]]]}]] // Reverse

Out[38]= or[equal[0, intersection[y, lb[x, y]]], not[member[x, CL]],
          subclass[y, image[x, set[APPLY[LEAST[x], y]]]]] == True

In[39]:= or[equal[0, intersection[y_, lb[x_, y_]]], not[member[x_, CL]],
          subclass[y_, image[x_, set[APPLY[LEAST[x_], y_]]]]] := True
```

Dual result:

Corollary. If **x** is a complete lattice and if **y** has a greatest element, then **APPLY[GREATEST[x], y]** is an upper bound for **y**.

```
In[40]:= Map[implies[member[x, CL], #] &, SubstTest[or,
           equal[0, intersection[y, lb[t, y]]], subclass[y, image[t, set[APPLY[LEAST[t], y]]]],
           not[member[t, CL]], t → inverse[x]]] // Reverse

Out[40]= or[equal[0, intersection[y, ub[x, y]]], not[member[x, CL]],
          subclass[y, image[inverse[x], set[APPLY[GREATEST[x], y]]]]] == True

In[41]:= or[equal[0, intersection[y_, ub[x_, y_]]], not[member[x_, CL]],
          subclass[y_, image[inverse[x_], set[APPLY[GREATEST[x_], y_]]]]] := True
```

The final step is to derive counterparts of the uniqueness theorems for complete lattices. A speedup by a factor of four resulted by omitting the following three proof steps in deriving the next theorem:

implies[and[p0, p1], p4], **implies[p1, p5]**, **implies[and[p1, p2, p3], p0]**.

No explicit hypothesis about the nature of the class **y** is needed. The derivation makes use of the fact that **y** is a subset of **fix[x]** is a consequence of the hypothesis that there is a least element **w**.

Theorem. If **x** is a complete lattice and **w ∈ least[x, y]**, then **w = APPLY[LEAST[x], y]**.

```
In[42]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p0], implies[and[p2, p3, p4, p5], p6],
           not[implies[and[p1, p2, p3], p6]], {p0 → subclass[y, fix[x]], p1 → member[x, CL],
            p2 → member[w, y], p3 -> subclass[y, image[x, set[w]]], p4 → member[y, V],
            p5 -> PARTIALORDER[x], p6 -> equal[w, APPLY[LEAST[x], y]]}]] // Reverse

Out[42]= or[equal[w, APPLY[LEAST[x], y]], not[member[w, y]],
           not[member[x, CL]], not[subclass[y, image[x, set[w]]]]] == True

In[43]:= or[equal[w_, APPLY[LEAST[x_], y_]], not[member[w_, y_]],
           not[member[x_, CL]], not[subclass[y_, image[x_, set[w_]]]]] := True
```

Restatement.

```
In[44]:= implies[and[member[x, CL], member[w, least[x, y]]], equal[w, APPLY[LEAST[x], y]]]

Out[44]= True
```

The dual result is obtained by simply replacing **x** with its inverse.

Theorem. If **x** is a complete lattice and **w ∈ greatest[x, y]**, then **w = APPLY[GREATEST[x], y]**.

```
In[45]:= Map[implies[member[x, CL], #] &,
           SubstTest[or, equal[w, APPLY[LEAST[t], y]], not[member[w, y]], not[member[t, CL]],
            not[subclass[y, image[t, set[w]]]], t → inverse[x]]] // Reverse

Out[45]= or[equal[w, APPLY[GREATEST[x], y]], not[member[w, y]],
           not[member[x, CL]], not[subclass[y, image[inverse[x], set[w]]]]] == True

In[46]:= or[equal[w_, APPLY[GREATEST[x_], y_]], not[member[w_, y_]],
           not[member[x_, CL]], not[subclass[y_, image[inverse[x_], set[w_]]]]] := True
```

Restatement.

```
In[47]:= implies[and[member[x, CL], member[w, greatest[x, y]]], equal[w, APPLY[GREATEST[x], y]]]

Out[47]= True
```