nonempty finite partial orders have maximal elements

Johan G. F. Belinfante
2009 May 4

```
In[1]:= SetDirectory["l:"]; << goedel.09may03a;<< tools.m

        :Package Title: goedel.09may03a            2009 May 3 at 5:45 p.m.

        It is now:  2009 May 4 at 13:53

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 April 6

        weightlimit = 40
```

summary

Any nonempty finite partial order has a maximal element. One can extend a finite partial order **p** to a total order, and then one can cut that down to a total order **t** with the properties that **p** ⊂ **t** and **fix[p]=fix[t]**. If **p** is not empty, then **t** is a non-empty finite total order, and so has a **t**-greatest element which is a maximal element for **p**.

derivation

Several wrappers are used to reduce the number of literals.

Lemma. If any greatest element of a nonempty total order is a maximal element of a partial order, then the partial order has a maximal element.

```
In[2]:= Map[implies[#, empty[to[fin[y]]]] &, SubstTest[and, subclass[u, v], empty[v],
           {u → fix[funpart[to[fin[y]]]], v → fix[funpart[po[x]]]}]] // Reverse

Out[2]= or[equal[0, to[fin[y]]], not[equal[0, funpart[po[x]]]],
          not[subclass[fix[funpart[to[fin[y]]]], fix[funpart[po[x]]]]]] == True

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

Lemma. An inclusion for **funpart**.

```
In[6]:= SubstTest[implies, subclass[p, t],
          subclass[intersection[domain[p], domain[funpart[t]]], domain[funpart[p]]],
          {p → po[x], t → to[fin[y]]}] // Reverse

Out[6]= or[not[subclass[po[x], to[fin[y]]]], subclass[
           intersection[fix[funpart[to[fin[y]]]], fix[po[x]]], fix[funpart[po[x]]]]] == True
```

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

Lemma. If a nonempty partial order can be extended to a finite total order with the same fixed-point class, then the partial order has a maximal member.

```
In[9]:= Map[not, SubstTest[and, implies[p2, p5], implies[and[p1, p2], p6],
          implies[and[p3, p5], p7], implies[and[p2, p4, p7], p8],
          implies[and[p2, p8], p9], not[implies[and[p1, p2, p3, p4], p9]],
          {p1 → not[equal[0, po[x]]], p2 → subclass[po[x], to[fin[y]]],
           p3 → equal[fix[po[x]], fix[to[fin[y]]]], p4 → equal[0, funpart[po[x]]],
           p5 → subclass[intersection[fix[funpart[to[fin[y]]]], fix[po[x]]],
             fix[funpart[po[x]]]], p6 → not[equal[0, to[fin[y]]]],
           p7 → subclass[fix[funpart[to[fin[y]]]], fix[funpart[po[x]]]],
           p8 → equal[0, to[fin[y]]], p9 → equal[0, po[x]]}]] // Reverse

Out[9]= or[equal[0, po[x]], not[equal[0, funpart[po[x]]]],
          not[equal[fix[po[x]], fix[to[fin[y]]]]], not[subclass[po[x], to[fin[y]]]]] == True

In[10]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

All of the wrappers can be removed.

Corollary.

```
In[11]:= SubstTest[implies, and[equal[x, po[u]], equal[y, to[fin[v]]]],
          or[equal[0, x], not[equal[0, funpart[x]]], not[equal[fix[x], fix[y]]],
           not[subclass[x, y]]], {u → x, v → y}] // Reverse

Out[11]= or[equal[0, x], not[equal[0, funpart[x]]],
          not[equal[fix[x], fix[y]]], not[member[y, FINITE]],
          not[PARTIALORDER[x]], not[subclass[x, y]], not[TOTALORDER[y]]] == True

In[12]:= or[equal[0, x_], not[equal[0, funpart[x_]]],
          not[equal[fix[x_], fix[y_]]], not[member[y_, FINITE]],
          not[PARTIALORDER[x_]], not[subclass[x_, y_]], not[TOTALORDER[y_]]] := True
```

The hypothesis **fix[x] = fix[y]** can be omitted and the finiteness hypothesis transferred to **x**. Some lemmas are needed.

Lemma. Any restriction to the fixed-point class of a finite relation is finite.

```
In[13]:= SubstTest[implies, and[subclass[u, v], member[v, FINITE]],
          member[u, FINITE], {u -> composite[id[fix[fin[x]]], y, id[fix[fin[x]]]],
           v → cartsq[fix[fin[x]]]}] // Reverse

Out[13]= member[composite[id[fix[fin[x]]], y, id[fix[fin[x]]]], FINITE] == True

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

The **fin** wrapper can be removed.

Corollary.

```
In[16]:= SubstTest[implies, equal[x, fin[z]],
           member[composite[id[fix[x]], y, id[fix[x]]], FINITE], z -> x] // Reverse

Out[16]= or[member[composite[id[fix[x]], y, id[fix[x]]], FINITE], not[member[x, FINITE]]] == True

In[17]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Comment: To shorten execution time, these proof steps are omitted in the following theorem: **implies[and[p0, p1], p3]**, **implies[and[p0, p2], p4]** and **implies[and[p0, p2], p5]**.

Theorem. If a nonempty finite partial order **x** is contained in a total order **y**, then it has a maximal element.

```
In[18]:= (Map[not,
            SubstTest[and, implies[and[p1, p3, p4, p5, p6], p7], implies[and[p0, p1, p2], p6],
              not[implies[and[p0, p1, p2], p7]], {p0 → equal[t, restrict[y, fix[x], fix[x]]],
               p1 → and[member[x, FINITE], PARTIALORDER[x], empty[funpart[x]]],
               p2 → and[subclass[x, y], TOTALORDER[y]], p3 → member[t, FINITE],
               p4 → TOTALORDER[t], p5 → equal[fix[x], fix[t]], p6 → subclass[x, t],
               p7 → empty[x]}] /. t -> restrict[y, fix[x], fix[x]]]) // Reverse

Out[18]= or[equal[0, x], not[equal[0, funpart[x]]], not[member[x, FINITE]],
           not[PARTIALORDER[x]], not[subclass[x, y]], not[TOTALORDER[y]]] == True

In[19]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The variable **y** can be eliminated.

Theorem. If a nonempty finite partial order **x** can be extended to a total order, then it has a maximal element.

```
In[20]:= Map[equal[V, #] &,
          SubstTest[class, y, implies[and[member[x, w], member[y, t]], not[empty[z]]],
           {w → intersection[FINITE, PO, complement[set[0]]],
            t → intersection[TO, image[S, set[x]]], z → funpart[x]}]]

Out[20]= or[equal[0, x], not[equal[0, funpart[x]]], not[member[x, FINITE]],
           not[member[x, image[inverse[S], TO]]], not[PARTIALORDER[x]]] == True

In[21]:= (% /. x → x_) /. Equal → SetDelayed
```

Any finite partial order can be extended to a total order. A variable-free statement of this fact is available in the **GOEDEL** program. The following theorem just introduces a variable to facilitate reasoning.

Theorem.

```
In[22]:= SubstTest[implies, and[member[x, y], subclass[y, z]], member[x, z],
           {y → intersection[FINITE, PO], z → image[inverse[S], TO]}] // Reverse

Out[22]= or[member[x, image[inverse[S], TO]],
           not[member[x, FINITE]], not[PARTIALORDER[x]]] == True

In[23]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Any nonempty finite partial order has a maximal element.

```
In[24]:= Map[not, SubstTest[and, implies[p1, p3],
            implies[and[p1, p2, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 → and[member[x, FINITE], PARTIALORDER[x]], p2 → empty[funpart[x]],
             p3 → member[x, image[inverse[S], TO]], p4 → empty[x]}]] // Reverse

Out[24]= or[equal[0, x], not[equal[0, funpart[x]]],
           not[member[x, FINITE]], not[PARTIALORDER[x]]] == True

In[25]:= or[equal[0, x_], not[equal[0, funpart[x_]]],
           not[member[x_, FINITE]], not[PARTIALORDER[x_]]] := True
```

Corollary. Restatement with wrappers.

```
In[26]:= SubstTest[or, equal[0, t], not[equal[0, funpart[t]]],
           not[member[t, FINITE]], not[PARTIALORDER[t]], t → po[fin[x]]] // Reverse

Out[26]= or[equal[0, po[fin[x]]], not[equal[0, funpart[po[fin[x]]]]]] == True

In[27]:= (% /. x → x_) /. Equal → SetDelayed
```

A better rewrite rule is possible.

Theorem.

```
In[28]:= equiv[equal[0, funpart[po[fin[x]]]], equal[0, po[fin[x]]]]

Out[28]= True

In[30]:= equal[0, funpart[po[fin[x_]]]] := equal[0, po[fin[x]]]
```

A variable-free statement can also be derived.

```
In[31]:= subclass[intersection[FINITE, PO, image[inverse[FUNPART], set[0]]], set[0]] //
          AssertTest

Out[31]= subclass[intersection[FINITE, PO, image[inverse[FUNPART], set[0]]], set[0]] == True

In[32]:= % /. Equal → SetDelayed
```

This can be strengthened to an equation and made into a rewrite rule.

```
In[33]:= equal[intersection[FINITE, PO, image[inverse[FUNPART], set[0]]], set[0]]

Out[33]= True

In[34]:= intersection[FINITE, PO, image[inverse[FUNPART], set[0]]] := set[0]
```