total suborder of a partial order

Johan G. F. Belinfante
2006 May 8

```
In[1]:= SetDirectory["l:"]; << goedel81.04a; << tools.m

        :Package Title: goedel81.04a          2006 May 4 at 2:10 p.m.

        It is now:  2006 May 8 at 16:34

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 May 8

        weightlimit = 40
```

summary

A total suborder of a partial order is a restriction.

derivation

Lemma 1.

```
In[2]:= Map[implies[subclass[#, to[x]],
            subclass[composite[id[fix[to[x]]], w, id[fix[to[x]]]], to[x]]] &,
          SubstTest[intersection, w, union[u, v], {u → to[x], v → inverse[to[x]]}]] // Reverse

Out[2]= or[not[subclass[intersection[w, inverse[to[x]]], to[x]]],
          subclass[composite[id[fix[to[x]]], w, id[fix[to[x]]]], to[x]]] == True

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

Lemma 2.

```
In[4]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
         {u → inverse[to[x]], v → inverse[po[y]], w → id[intersection[inverse[to[x]], po[y]]]}]

Out[4]= or[not[subclass[to[x], po[y]]], subclass[intersection[inverse[to[x]], po[y]],
           id[intersection[fix[po[y]], fix[to[x]]]]]] == True

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

If a total order **to[x]** is a subclass of a partial order **po[y]**, then the restriction of **po[y]** to **fix[to[x]]** is a subclass of **to[x]**.

```
In[6]:= Map[not, SubstTest[and, implies[p1, p2],
          implies[p2, p3], implies[p3, p4], not[implies[p1, p4]],
          {p1 → subclass[to[x], po[y]], p2 -> subclass[intersection[inverse[to[x]], po[y]],
             id[intersection[fix[po[y]], fix[to[x]]]]],
           p3 → subclass[intersection[inverse[to[x]], po[y]], to[x]],
           p4 → subclass[restrict[po[y], fix[to[x]], fix[to[x]]], to[x]]}]]

Out[6]= or[not[subclass[to[x], po[y]]],
          subclass[composite[id[fix[to[x]]], po[y], id[fix[to[x]]]], to[x]]] == True

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

Theorem. A total suborder of a partial order is a restriction.

```
In[8]:= Map[implies[subclass[to[x], po[y]], #] &,
          SubstTest[and, implies[p, subclass[u, v]], subclass[v, u], {p -> subclass[to[x], po[y]],
            u -> composite[id[fix[to[x]]], po[y], id[fix[to[x]]]], v -> to[x]}]] // Reverse

Out[8]= or[equal[composite[id[fix[to[x]]], po[y], id[fix[to[x]]]], to[x]],
          not[subclass[to[x], po[y]]]] == True

In[9]:= or[equal[composite[id[fix[to[x_]]], po[y_], id[fix[to[x_]]]], to[x_]],
          not[subclass[to[x_], po[y_]]]] := True
```

---

eliminating the variable x

The first step is to removed the **to** wrapper.

```
In[10]:= SubstTest[implies, and[equal[x, to[u]], subclass[x, po[y]]],
          equal[x, restrict[po[y], fix[x], fix[x]]], u → x]

Out[10]= or[equal[x, composite[id[fix[x]], po[y], id[fix[x]]]],
           not[subclass[x, po[y]]], not[TOTALORDER[x]]] == True

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

The following lemma reformulates the conclusion.

```
In[12]:= SubstTest[implies,
          and[member[pair[u, v], composite[Id, w]], member[u, t]], member[v, image[w, t]],
          {t → image[CART, Id], u → cart[fix[x], fix[x]], v → x, w → IMAGE[id[po[y]]]}]

Out[12]= or[member[x, image[IMAGE[id[po[y]]], image[CART, Id]]],
           not[equal[x, composite[id[fix[x]], po[y], id[fix[x]]]]], not[member[x, V]]] == True

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

```
In[14]:= Map[not,
          SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 → member[x, intersection[P[po[y]], TO]],
            p2 -> equal[x, composite[id[fix[x]], po[y], id[fix[x]]]],
            p3 → member[x, image[IMAGE[id[po[y]]], image[CART, Id]]]}]]

Out[14]= or[member[x, image[IMAGE[id[po[y]]], image[CART, Id]]],
           not[member[x, V]], not[subclass[x, po[y]]], not[TOTALORDER[x]]] == True

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

Now the variable **x** is eliminated:

```
In[16]:= Map[equal[V, #] &, SubstTest[class, x,
             implies[member[x, u], member[x, v]], {u -> intersection[P[po[y]], TO],
              v -> image[IMAGE[id[po[y]]], image[CART, Id]]}]] // Reverse

Out[16]= subclass[intersection[TO, P[po[y]]], image[IMAGE[id[po[y]]], image[CART, Id]]] == True

In[17]:= (% /. y → y_) /. Equal → SetDelayed
```

Lemma.

```
In[18]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → intersection[x, image[IMAGE[id[y]], z]], v -> image[IMAGE[id[y]], z], w → P[y]}]

Out[18]= subclass[U[intersection[x, image[IMAGE[id[y]], z]]], y] == True

In[19]:= subclass[U[intersection[x_, image[IMAGE[id[y_]], z_]]], y_] := True
```

Theorem.

```
In[20]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → intersection[TO, P[po[x]]],
             v → intersection[TO, image[IMAGE[id[po[x]]], image[CART, Id]]]}] // Reverse

Out[20]= equal[intersection[TO, image[IMAGE[id[po[x]]], image[CART, Id]]],
           intersection[TO, P[po[x]]]] == True
```

It is not entirely clear how best to orient this equation as a rewrite rule. Tentatively the following will be adopted.

```
In[21]:= intersection[TO, image[IMAGE[id[po[x_]]], image[CART, Id]]] :=
          intersection[TO, P[po[x]]]
```