cover relation of a total order

Johan G. F. Belinfante
2010 February 11

```
In[1]:= SetDirectory["l:"]; << goedel.10feb10a;<< tools.m
        :Package Title: goedel.10feb10a                2010 February 10 at 2:00 p.m.
        It is now:  2010 Feb 11 at 13:57
        Loading Simplification Rules
        TOOLS.M                                    Revised 2010 January 29
        weightlimit = 40
```

summary

Rewrite rules about the cover relation of a total order are derived.

simplification rules

Some basic simplification rules are derived in this section. Most of these rewrite rules were suggested by the work done in the next section, but only a little of this is actually needed there.

Theorem.

```
In[2]:= SubstTest[subclass, domain[cover[t]], domain[t], t → to[x]] // Reverse
Out[2]= subclass[domain[cover[to[x]]], fix[to[x]]] == True
```

```
In[3]:= subclass[domain[cover[to[x_]]], fix[to[x_]]] := True
```

Corollary.

```
In[4]:= equal[composite[cover[to[x]], id[fix[to[x]]]], cover[to[x]]]
Out[4]= True
```

```
In[5]:= composite[cover[to[x_]], id[fix[to[x_]]]] := cover[to[x]]
```

Comment. The dual counterpart of the above rewrite rule is already available.

Theorem.

```
In[6]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
          {u → cover[to[x]], v → to[x], w → to[x]}] // Reverse
```

```
Out[6]= subclass[composite[cover[to[x]], to[x]], to[x]] == True
```

```
In[7]:= subclass[composite[cover[to[x_]], to[x_]], to[x_]] := True
```

Theorem.

```
In[8]:= SubstTest[implies, subclass[u, v], subclass[composite[t, u], composite[t, v]],
          {t → to[x], u → cover[to[x]], v → to[x], w → to[x]}] // Reverse
```

```
Out[8]= subclass[composite[to[x], cover[to[x]]], to[x]] == True
```

```
In[9]:= subclass[composite[to[x_], cover[to[x_]]], to[x_]] := True
```

Theorem.

```
In[10]:= SubstTest[implies, and[subclass[u, v], empty[image[t, v]]],
           empty[image[t, u]], {t -> composite[SECOND, id[inverse[to[x]]]],
            u → cover[to[x]], v → intersection[Di, to[x]]}] // Reverse
```

```
Out[10]= equal[0, fix[composite[cover[to[x]], to[x]]]] == True
```

```
In[11]:= fix[composite[cover[to[x_]], to[x_]]] := 0
```

Corollary.

```
In[12]:= SubstTest[implies, empty[fix[composite[u, v]]],
           empty[fix[composite[v, u]]], {u -> cover[to[x]], v -> to[x]}] // Reverse
```

```
Out[12]= equal[0, fix[composite[to[x], cover[to[x]]]]] == True
```

```
In[13]:= fix[composite[to[x_], cover[to[x_]]]] := 0
```

---

sharpening an existing rule

In this section an existing rewrite rule is sharpened, replacing an inclusion by an equation. The main theorem at the end of this section was suggested by results about well-ordering learned in the course of studying the material in pages 75-76 in the following reference, but it must be pointed out that the treatment given by Patrick Suppes uses strict orders rather than reflexive orders, so some work would be needed to see the connection.

```
"Patrick Suppes, Axiomatic Set Theory, Dover Publications, New York, 1972.";
```

Lemma.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → member[pair[x, y], cover[to[z]]],
            p2 → member[x, fix[to[z]]], p3 → member[pair[x, x], to[z]]}]] // Reverse
```

```
Out[14]= or[member[pair[x, x], to[z]], not[member[pair[x, y], cover[to[z]]]]] == True
```

```
In[15]:= or[member[pair[x_, x_], to[z_]], not[member[pair[x_, y_], cover[to[z_]]]]] := True
```

Technical Lemma.

```
In[16]:= or[and[member[pair[x, x], to[z]], member[pair[x, y], to[z]]],
           and[member[pair[x, x], to[z]], not[member[y, fix[to[z]]]]],
           and[member[pair[x, y], to[z]], not[member[x, V]]],
           and[not[member[x, V]], not[member[y, fix[to[z]]]]],
           not[member[pair[x, y], cover[to[z]]]]] // NotNotTest
```

```
Out[16]= or[and[member[pair[x, x], to[z]], member[pair[x, y], to[z]]],
           and[member[pair[x, x], to[z]], not[member[y, fix[to[z]]]]],
           and[member[pair[x, y], to[z]], not[member[x, V]]],
           and[not[member[x, V]], not[member[y, fix[to[z]]]]],
           not[member[pair[x, y], cover[to[z]]]]] == True
```

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

Theorem. (Sharpening an available inclusion to an equation.)

```
In[18]:= SubstTest[and, implies[p, subclass[u, v]],
           implies[p, subclass[v, u]], {p -> member[pair[x, y], cover[to[z]]],
            u -> image[to[z], set[x]], v -> union[image[to[z], set[y]], set[x]]}]
```

```
Out[18]= or[equal[image[to[z], set[x]], union[image[to[z], set[y]], set[x]]],
           not[member[pair[x, y], cover[to[z]]]]] == True
```

```
In[19]:= or[equal[image[to[z_], set[x_]], union[image[to[z_], set[y_]], set[x_]]],
           not[member[pair[x_, y_], cover[to[z_]]]]] := True
```

---

eliminating a variable

The cover relation of a total order is a function, and accordingly the variable **y** in the statement **pair[x, y] ∈ cover[to[z]]** is uniquely determined by **x**. This permits one to eliminate this variable.

Theorem.

```
In[20]:= SubstTest[image, funpart[t], set[y], t → cover[to[x]]] // Reverse
```

```
Out[20]= image[cover[to[x]], set[y]] == set[APPLY[cover[to[x]], y]]
```

```
In[21]:= image[cover[to[x_]], set[y_]] := set[APPLY[cover[to[x]], y]]
```

Lemma. (Eliminating a variable.)

```
In[22]:= Map[equal[V, #] &,
           SubstTest[class, w, or[not[member[pair[y, w], s]], subclass[image[t, set[y]],
              union[image[t, set[w]], set[y]]]], {s → cover[to[x]], t → to[x]}]]
```

```
Out[22]= or[not[member[y, domain[cover[to[x]]]]], subclass[image[to[x], set[y]],
             union[image[to[x], set[APPLY[cover[to[x]], y]]], set[y]]]] == True
```

```
In[23]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Here too the inclusion can be sharpened to an equation. This will now be done.

Lemma.

```
In[24]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
           {u -> union[id[fix[to[x]]], composite[to[x], cover[to[x]]]],
            v -> to[x], w → set[y]}] // Reverse

Out[24]= or[member[pair[y, APPLY[cover[to[x]], y]], to[x]],
           not[member[APPLY[cover[to[x]], y], fix[to[x]]]]] == True
```

```
In[25]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[26]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
              not[implies[p1, p3]], {p1 -> member[x, domain[cover[to[y]]]],
               p2 → member[x, fix[to[y]]], p3 → member[pair[x, x], to[y]]}]] // Reverse

Out[26]= or[member[pair[x, x], to[y]], not[member[x, domain[cover[to[y]]]]]] == True
```

```
In[27]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. (Sharpening an inclusion to an equation.)

```
In[31]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
           {p -> member[y, domain[cover[to[x]]]], u -> image[to[x], set[y]],
            v -> union[image[to[x], set[APPLY[cover[to[x]], y]]], set[y]]}]

Out[31]= or[equal[image[to[x], set[y]],
              union[image[to[x], set[APPLY[cover[to[x]], y]]], set[y]]],
            not[member[y, domain[cover[to[x]]]]]] == True
```

```
In[33]:= or[equal[image[to[x_], set[y_]],
              union[image[to[x_], set[APPLY[cover[to[x_]], y_]]], set[y_]]],
            not[member[y_, domain[cover[to[x_]]]]]] := True
```

removing the to[x] wrapper

The **to[x]** wrappers are removed from the main results of the preceding two sections, as well as from another rewrite rule derived elsewhere.

Theorem. If an element **x** is covered by an element **y** with respect to a total order **z**, then the vertical section of **z** at **x** is equal to the union of the singleton of **x** and the vertical section of **z** at **y**.

```
In[34]:= SubstTest[implies, equal[z, to[t]], or[not[member[pair[x, y], cover[z]]],
           equal[image[z, set[x]], union[image[z, set[y]], set[x]]]], t → z] // Reverse

Out[34]= or[equal[image[z, set[x]], union[image[z, set[y]], set[x]]],
          not[member[pair[x, y], cover[z]]], not[TOTALORDER[z]]] == True

In[35]:= or[equal[image[z_, set[x_]], union[image[z_, set[y_]], set[x_]]],
          not[member[pair[x_, y_], cover[z_]]], not[TOTALORDER[z_]]] := True
```

Theorem. Eliminating the **to[z]** wrapper from a converse result derived elsewhere.

```
In[36]:= SubstTest[implies, equal[z, to[t]],
          or[equal[x, y], member[pair[x, y], cover[z]], not[member[pair[x, y], z]],
           not[subclass[image[z, set[x]], union[image[z, set[y]], set[x]]]]], t → z] // Reverse

Out[36]= or[equal[x, y], member[pair[x, y], cover[z]], not[member[pair[x, y], z]],
          not[subclass[image[z, set[x]], union[image[z, set[y]], set[x]]]],
          not[TOTALORDER[z]]] == True

In[37]:= or[equal[x_, y_], member[pair[x_, y_], cover[z_]], not[member[pair[x_, y_], z_]],
          not[subclass[image[z_, set[x_]], union[image[z_, set[y_]], set[x_]]]],
          not[TOTALORDER[z_]]] := True
```

Theorem. If there is an element that covers **y** with respect to a total order **x**, then the vertical section of **x** at **y** is the union of the singleton of **y** and the vertical section of **x** at the unique element **APPLY[cover[x], y]** that covers **y**.

```
In[38]:= SubstTest[implies, equal[x, to[t]],
          or[equal[image[x, set[y]], union[image[x, set[APPLY[cover[x], y]]], set[y]]],
           not[member[y, domain[cover[x]]]]], t → x] // Reverse

Out[38]= or[equal[image[x, set[y]], union[image[x, set[APPLY[cover[x], y]]], set[y]]],
          not[member[y, domain[cover[x]]]], not[TOTALORDER[x]]] == True

In[39]:= or[equal[image[x_, set[y_]], union[image[x_, set[APPLY[cover[x_], y_]]], set[y_]]],
          not[member[y_, domain[cover[x_]]]], not[TOTALORDER[x_]]] := True
```