image[COVER, PO]

Johan G. F. Belinfante
2010 July 6

```
In[1]:= SetDirectory["l:"]; << goedel.10jul05a; << tools.m

        :Package Title: goedel.10jul05a              2010 July 5 at 3:10 p.m.

        It is now:  2010 Jul 6 at 11:53

        Loading Simplification Rules

        TOOLS.M                                    Revised 2010 February 26

        weightlimit = 40
```

summary

This notebook is about characterizing cover relations of partial orders. See pages 20-21 in this book:

```
In[2]:= "Egbert Harzheim, Ordered Sets, Advances in Mathematics, volume 7, Springer
        Science+Business Media, Inc., 2005.  ISBN 0387-24219-8.  QA171.48 .H37";
```

Recall that the **cover relation** of a relation **x** is defined as follows:

```
In[3]:= dif[intersection[Di, x], composite[intersection[Di, x], intersection[Di, x]]]

Out[3]= cover[x]
```


acyclic

A relation **x** is **acyclic** if **fix[trv[x]] = 0**. The cover relation of a partial order is acyclic. In this section, a variable-free statement this fact is derived.

The **GOEDEL** program already has a rewrite rule asserting that **cover[po[x]]** is acyclic. The first step will be to eliminate the **po[x]** wrapper.

Theorem. A wrapper-free statement.

```
In[4]:= SubstTest[implies, equal[x, po[t]], empty[fix[trv[cover[x]]]], t → x] // Reverse

Out[4]= or[equal[0, fix[trv[cover[x]]]], not[PARTIALORDER[x]]] = True

In[5]:= or[equal[0, fix[trv[cover[x_]]]], not[PARTIALORDER[x_]]] := True
```

Technical lemma.

```
In[6]:= implies[member[x, V],
          member[x, complement[dif[PO, image[inverse[COVER], ACYCLIC]]]]] // NotNotTest

Out[6]= or[and[equal[0, fix[trv[cover[x]]]], member[cover[x], V]],
          not[member[x, V]], not[PARTIALORDER[x]]] == True

In[7]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem. The cover relation of a partial order is acyclic.

```
In[8]:= Map[equal[V, #] &, SubstTest[class, x, implies[member[x, V], member[x, t]],
          t -> complement[dif[PO, image[inverse[COVER], ACYCLIC]]]]]

Out[8]= subclass[image[COVER, PO], ACYCLIC] == True

In[9]:= subclass[image[COVER, PO], ACYCLIC] := True
```

Corollary. The same is true for total orders.

```
In[10]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → image[COVER, TO], v → image[COVER, PO], w → ACYCLIC}] // Reverse

Out[10]= subclass[image[COVER, TO], ACYCLIC] == True

In[11]:= subclass[image[COVER, TO], ACYCLIC] := True
```

Comment. In general, the cover relation of an arbitrary relation need not be acyclic. For example, the cyclic function which interchanges **0** and **1** is its own cover relation.

---

totally intransitive

A relation **x** is **totally intransitive** if it is acyclic and $\mathbf{x} \cap (\mathbf{trv[x]} \circ \mathbf{trv[x]}) = \mathbf{0}$. The class of relations satisfying the latter condition is **fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]**.

```
In[12]:= member[x, fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]]

Out[12]= and[equal[0, intersection[x, composite[trv[x], trv[x]]]],
           member[x, V], subclass[x, cart[V, V]]]
```

In this section it is shown that the cover relation of a partial order is totally intransitive.

Lemma.

```
In[13]:= SubstTest[implies, subclass[u, v], subclass[composite[u, u], composite[v, v]],
           {u → trv[cover[po[x]]], v → intersection[Di, po[x]]}] // Reverse

Out[13]= subclass[composite[trv[cover[po[x]]], trv[cover[po[x]]]],
           composite[intersection[Di, po[x]], intersection[Di, po[x]]]] == True
```

```
In[14]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. The cover relation of **po[x]** is totally intransitive.

```
In[15]:= Map[implies[#, empty[
            intersection[composite[trv[cover[po[x]]], trv[cover[po[x]]]], cover[po[x]]]] &,
          SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → id[cover[po[x]]], u -> composite[trv[cover[po[x]]], trv[cover[po[x]]]],
            v -> composite[intersection[Di, po[x]], intersection[Di, po[x]]]}]]
```

```
Out[15]= equal[0,
           intersection[composite[trv[cover[po[x]]], trv[cover[po[x]]]], cover[po[x]]]] == True
```

```
In[16]:= intersection[composite[trv[cover[po[x_]]], trv[cover[po[x_]]]], cover[po[x_]]] := 0
```

A variable-free formulation will now be derived using **reify**. This can be done without having to first remove the **po** wrapper.

Lemma. A sethood rule.

```
In[17]:= SubstTest[member, cover[setpart[t]], V, t → po[setpart[x]]] // Reverse
```

```
Out[17]= member[cover[po[setpart[x]]], V] == True
```

```
In[18]:= member[cover[po[setpart[x_]]], V] := True
```

Theorem. A variable-free statement.

```
In[19]:= Map[empty, SubstTest[reify, x, dif[set[po[setpart[x]]], t],
           t -> image[inverse[COVER], fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]]]]
```

```
Out[19]= subclass[image[COVER, PO], fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]] == True
```

```
In[20]:= subclass[image[COVER, PO], fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]] := True
```

---

the reverse direction

In this section an inclusion in the opposite direction is derived.

Lemma.

```
In[21]:= SubstTest[implies, member[u, v], member[cover[u], image[COVER, v]],
           {u → union[id[udora[x]], trv[x]], v → PO}] // Reverse
```

```
Out[21]= or[member[cover[trv[x]], image[COVER, PO]],
           not[member[domain[x], V]], not[member[range[x], V]],
           not[PARTIALORDER[union[id[union[domain[x], range[x]]], trv[x]]]]] == True
```

```
In[22]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. If **x** is acyclic, then **cover[trv[x]] ∈ image[COVER, PO]**.

```
In[23]:= Map[implies[member[x, y], not[#]] &, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[p1, p4], implies[and[p2, p3, p4], p5],
           not[implies[p1, p5]], {p1 → and[equal[0, fix[trv[x]]], member[x, V]],
            p2 -> member[domain[x], V], p3 -> member[range[x], V],
            p4 -> PARTIALORDER[union[id[union[domain[x], range[x]]], trv[x]]],
            p5 -> member[cover[trv[x]], image[COVER, PO]]}]] // Reverse

Out[23]= or[member[cover[trv[x]], image[COVER, PO]],
           not[equal[0, fix[trv[x]]]], not[member[x, y]]] == True

In[24]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary.

```
In[25]:= SubstTest[or, member[cover[trv[t]], image[COVER, PO]],
           not[equal[0, fix[trv[t]]]], not[member[t, V]], t → setpart[x]] // Reverse

Out[25]= or[member[cover[trv[setpart[x]]], image[COVER, PO]],
           not[equal[0, fix[trv[setpart[x]]]]]] == True

In[26]:= (% /. x → x_) /. Equal → SetDelayed
```

The formula for **cover[x]** can be simplified when **x** is irreflexive:

```
In[27]:= implies[subclass[x, Di], equal[cover[x], dif[x, composite[x, x]]]]

Out[27]= True
```

A special case of this is needed.

Lemma. Formula for **cover[trv[x]]** for the special case of an acyclic relation.

```
In[28]:= SubstTest[implies, subclass[t, Di],
           equal[cover[t], intersection[t, complement[composite[t, t]]]], t → trv[x]] // Reverse

Out[28]= or[equal[cover[trv[x]], intersection[complement[composite[trv[x], trv[x]]], trv[x]]],
           not[equal[0, fix[trv[x]]]]] == True

In[29]:= or[equal[cover[trv[x_]],
            intersection[complement[composite[trv[x_], trv[x_]]], trv[x_]]],
           not[equal[0, fix[trv[x_]]]]] := True
```

This result will now be used to show that any totally intransitive relation is the cover relation of a partial order.

Lemma.

```
In[30]:= SubstTest[implies, and[disjoint[u, v], equal[w, union[u, v]]], equal[u, dif[w, v]],
           {u → composite[Id, x], v → composite[trv[x], trv[x]], w → trv[x]}] // Reverse

Out[30]= or[equal[composite[Id, x],
            intersection[complement[composite[trv[x], trv[x]]], trv[x]]],
           not[equal[0, intersection[x, composite[trv[x], trv[x]]]]]] == True
```

```
In[31]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. If **x** is totally intransitive, then **x = cover[trv[x]]**.

```
In[32]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 → empty[fix[trv[x]]], p2 → disjoint[x, composite[trv[x], trv[x]]], p3 -> equal[
              cover[trv[x]], intersection[complement[composite[trv[x], trv[x]]], trv[x]]],
            p4 -> equal[composite[Id, x], intersection[complement[composite[trv[x], trv[x]]],
               trv[x]]], p5 -> equal[composite[Id, x], cover[trv[x]]]}] // Reverse

Out[32]= or[equal[composite[Id, x], cover[trv[x]]], not[equal[0, fix[trv[x]]]],
           not[equal[0, intersection[x, composite[trv[x], trv[x]]]]]] == True

In[33]:= or[equal[composite[Id, x_], cover[trv[x_]]], not[equal[0, fix[trv[x_]]]],
           not[equal[0, intersection[composite[trv[x_], trv[x_]], x_]]]] := True
```

Corollary.

```
In[34]:= Map[not,
           SubstTest[and, implies[and[p1, p2], p3], implies[p1, p4], implies[and[p3, p4], p5],
            not[implies[and[p1, p2], p5]], {p1 → empty[fix[trv[setpart[x]]]],
             p2 → disjoint[setpart[x], composite[trv[setpart[x]], trv[setpart[x]]]],
             p3 → equal[composite[Id, setpart[x]], cover[trv[setpart[x]]]],
             p4 → member[cover[trv[setpart[x]]], image[COVER, PO]],
             p5 → member[composite[Id, setpart[x]], image[COVER, PO]]}]] // Reverse

Out[34]= or[member[composite[Id, setpart[x]], image[COVER, PO]],
           not[equal[0, fix[trv[setpart[x]]]]], not[equal[0,
             intersection[composite[trv[setpart[x]], trv[setpart[x]]], setpart[x]]]]] == True

In[35]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (Removing the **composite[Id, setpart[x]]** wrapper.)

```
In[36]:= SubstTest[implies, and[equal[x, composite[Id, setpart[t]]], member[x, ACYCLIC],
            member[x, fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]]],
           member[x, image[COVER, PO]], t → x] // Reverse

Out[36]= or[member[x, image[COVER, PO]], not[equal[0, fix[trv[x]]]],
           not[equal[0, intersection[x, composite[trv[x], trv[x]]]]],
           not[member[x, V]], not[subclass[x, cart[V, V]]]] == True

In[37]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (Eliminating the variable **x**.)

```
In[38]:= Map[equal[V, #] &, SubstTest[class, x, implies[member[x, u], member[x, v]],
           {u → intersection[ACYCLIC, fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]],
            v → image[COVER, PO]}]]

Out[38]= subclass[intersection[ACYCLIC, fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]],
           image[COVER, PO]] == True
```

```
In[39]:= % /. Equal → SetDelayed
```

Combining these inclusions yields an equation that can be made into a rewrite rule. How best to orient this rule is not obvious. The choice made here has the advantage of replacing a complex expression with a simpler one, but the disadvantage that in order to use the rule, one must already be aware of its existence.

Main Theorem. A relation is the cover relation of a partial order if and only if it is totally intransitive.

```
In[40]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → intersection[ACYCLIC, fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]],
           v → image[COVER, PO]}]

Out[40]= equal[image[COVER, PO],
           intersection[ACYCLIC, fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]]] == True

In[41]:= intersection[ACYCLIC, fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]] :=
          image[COVER, PO]
```

corollaries

Had the rewrite rule for the main theorem been turned around, the rewrite rules derived in this section would be unnecessary, but then the simple statement **x ∈ image[COVER, PO]** would invariably be expanded out as a fairly complex expression. By rendering these rewrite rules as statements of fact, one has much more control over complexity. On the other hand, in order to make any use of these results, one must already know about them. If one were using an automated reasoning program such as **Otter** this is no disadvantage because such programs use resolution to automatically access all available theorems.

Lemma. A general inclusion.

```
In[42]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → image[COVER, x], v → fix[COVER], w → P[cart[V, V]]}] // Reverse

Out[42]= subclass[U[image[COVER, x]], cart[V, V]] == True

In[43]:= subclass[U[image[COVER, x_]], cart[V, V]] := True
```

Theorem. Cover relations are relations.

```
In[44]:= SubstTest[implies, and[member[x, u], subclass[u, v]],
           member[x, v], {u → image[COVER, y], v → P[cart[V, V]]}] // Reverse

Out[44]= or[not[member[x, image[COVER, y]]], subclass[x, cart[V, V]]] == True

In[45]:= or[not[member[x_, image[COVER, y_]]], subclass[x_, cart[V, V]]] := True
```

Theorem. The cover relation of any partial order is acyclic.

```
In[46]:= SubstTest[implies, and[member[x, u], subclass[u, v]],
           member[x, v], {u → image[COVER, PO], v → ACYCLIC}] // MapNotNot // Reverse

Out[46]= or[equal[0, fix[trv[x]]], not[member[x, image[COVER, PO]]]] == True

In[47]:= or[equal[0, fix[trv[x_]]], not[member[x_, image[COVER, PO]]]] := True
```

Theorem. The cover relation of any partial order is totally intransitive.

```
In[48]:= Map[implies[member[x, image[COVER, PO]], #] &, SubstTest[member, x, intersection[y, z],
           {y → ACYCLIC, z -> fix[composite[DISJOINT, COMPOSE, DUP, HULL[TRV]]]}]] // MapNotNot

Out[48]= or[equal[0, intersection[x, composite[trv[x], trv[x]]]],
           not[member[x, image[COVER, PO]]]] == True

In[49]:= or[equal[0, intersection[x_, composite[trv[x_], trv[x_]]]],
           not[member[x_, image[COVER, PO]]]] := True
```