cover[cover[x]]

Johan G. F. Belinfante
2010 January 28

```
In[1]:= SetDirectory["l:"]; << goedel.10jan27a; << tools.m

        :Package Title: goedel.10jan27a              2010 January 27 at 3:00 p.m.

        It is now:  2010 Jan 28 at 12:0

        Loading Simplification Rules

        TOOLS.M                                    Revised 2010 January 7

        weightlimit = 40
```

summary

The cover relation of the cover relation of a relation is equal to the cover relation. In other words, the constructor **cover** is idempotent.

special case of K

The general case will be patterned on the special case of **K**, the cover relation of the subset relation. This special case could of course be recovered from the general case, but the formulas are easier to follow in the special case, so it is perhaps instructive to do this case first.

Lemma.

```
In[7]:= SubstTest[implies, subclass[x, y],
          subclass[composite[x, x], composite[y, y]], {x → K, y → PS}] // Reverse

Out[7]= subclass[composite[K, K], composite[PS, PS]] == True

In[8]:= % /. Equal → SetDelayed
```

Lemma.

```
In[13]:= SubstTest[implies, and[disjoint[x, z], subclass[y, z]],
           disjoint[x, y], {x → K, y → composite[K, K], z → composite[PS, PS]}] // Reverse

Out[13]= equal[0, intersection[K, composite[K, K]]] == True

In[16]:= intersection[K, composite[K, K]] := 0
```

Theorem.

```
In[19]:= Map[equal[#, K] &, SubstTest[dif, intersection[Di, x],
           composite[intersection[Di, x], intersection[Di, x]], x → K]]

Out[19]= equal[K, cover[K]] == True

In[21]:= cover[K] := K
```

---

general case

Lemma.

```
In[23]:= SubstTest[implies, subclass[u, v], subclass[composite[u, u], composite[v, v]],
           {u → cover[x], v → intersection[Di, x]}] // Reverse

Out[23]= subclass[composite[cover[x], cover[x]],
           composite[intersection[Di, x], intersection[Di, x]]] == True

In[24]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[27]:= SubstTest[implies, and[disjoint[u, w], subclass[v, w]],
           disjoint[u, v], {u → cover[x], v → composite[cover[x], cover[x]],
            w → composite[intersection[Di, x], intersection[Di, x]]}] // Reverse

Out[27]= equal[0, intersection[composite[cover[x], cover[x]], cover[x]]] == True

In[29]:= intersection[composite[cover[x_], cover[x_]], cover[x_]] := 0
```

Lemma. Simplification rule.

```
In[35]:= equal[intersection[Di, cover[x]], cover[x]]

Out[35]= True

In[37]:= intersection[Di, cover[x_]] := cover[x]
```

Theorem.

```
In[38]:= Map[equal[#, cover[x]] &, SubstTest[dif, intersection[Di, t],
           composite[intersection[Di, t], intersection[Di, t]], t → cover[x]]]

Out[38]= equal[cover[x], cover[cover[x]]] == True

In[40]:= cover[cover[x_]] := cover[x]
```

other special cases

Theorem.

```
In[51]:= SubstTest[cover, cover[t], t → restrict[S, omega, omega]] // Reverse
Out[51]= cover[composite[id[omega], SUCC]] == composite[id[omega], SUCC]

In[52]:= cover[composite[id[omega], SUCC]] := composite[id[omega], SUCC]
```

Theorem.

```
In[53]:= SubstTest[cover, cover[t], t → restrict[S, OMEGA, OMEGA]] // Reverse
Out[53]= cover[composite[id[OMEGA], SUCC]] == composite[id[OMEGA], SUCC]

In[54]:= cover[composite[id[OMEGA], SUCC]] := composite[id[OMEGA], SUCC]
```

Theorem.

```
In[57]:= SubstTest[cover, cover[t], t → composite[id[P[x]], PS]] // Reverse
Out[57]= cover[composite[id[P[x]], K]] == composite[id[P[x]], K]

In[58]:= cover[composite[id[P[x_]], K]] := composite[id[P[x]], K]
```