fix[composite[Q,K]]

Johan G. F. Belinfante
2007 July 8

```
In[1]:= SetDirectory["l:"]; << goedel95.07a; << tools.m

        :Package Title: goedel95.07a      2007 July 7 at 4:05 a.m.

        It is now:  2007 Jul 8 at 10:18

        Loading Simplification Rules

        TOOLS.M                                      Revised 2007 June 25

        weightlimit = 40
```

summary

A finite set cannot be equipollent to one that covers it. The class of sets equipollent to one that covers it is invariant under equipollence. Comment. It does not matter if one works with the cover relation of its inverse.

```
In[2]:= fix[composite[Q, inverse[K]]]

Out[2]= fix[composite[Q, K]]
```

All the results in this notebook are valid independently of the axiom of regularity and the axiom of choice.

derivation

Theorem.

```
In[3]:= SubstTest[fix, composite[id[x], y], {x → FINITE, y → composite[Q, K]}]

Out[3]= intersection[FINITE, fix[composite[Q, K]]] == 0

In[4]:= intersection[FINITE, fix[composite[Q, K]]] := 0
```

Theorem. A stronger result: one can replace **FINITE** with **DEDEKIND**.

```
In[5]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
          {t -> composite[FIRST, id[Q]], u → K, v → PS}] // Reverse

Out[5]= equal[0, intersection[DEDEKIND, fix[composite[Q, K]]]] == True

In[6]:= intersection[DEDEKIND, fix[composite[Q, K]]] := 0
```

SUCC results

In many applications, one may wish to replace **K** with **SUCC**. This can be done if one is dealing with regular classes, or more generally with sets that do not belong to themselves. The basic connection is this:

```
In[7]:= subclass[SUCC, union[K, id[fix[E]]]]

Out[7]= True
```

Theorem.

```
In[8]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
          {t -> composite[id[RUSSELL], FIRST, id[Q]], u → SUCC, v → union[K, id[fix[E]]]}] //
         Reverse

Out[8]= subclass[intersection[RUSSELL, fix[composite[Q, SUCC]]], fix[composite[Q, K]]] == True

In[9]:= subclass[intersection[RUSSELL, fix[composite[Q, SUCC]]], fix[composite[Q, K]]] := True
```

Corollary.

```
In[10]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> intersection[RUSSELL, fix[composite[Q, SUCC]]],
            v -> fix[composite[Q, K]], w → complement[FINITE]}] // Reverse

Out[10]= equal[0, intersection[FINITE, RUSSELL, fix[composite[Q, SUCC]]]] == True

In[11]:= intersection[FINITE, RUSSELL, fix[composite[Q, SUCC]]] := 0
```

Corollary. A similar result, using **DEDEKIND** in place of **FINITE**.

```
In[12]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> intersection[RUSSELL, fix[composite[Q, SUCC]]],
            v -> fix[composite[Q, K]], w → complement[DEDEKIND]}] // Reverse

Out[12]= equal[0, intersection[DEDEKIND, RUSSELL, fix[composite[Q, SUCC]]]] == True

In[13]:= intersection[DEDEKIND, RUSSELL, fix[composite[Q, SUCC]]] := 0
```

an example

The condition **subclass[P[x], succ[x]]** was studied in connection with checking whether certain low rank sets satisfy the condition that subsets of members are members. For sets satisfying this condition, the following establishes that such sets do not belong to themselves. (Of course, for sets that have rank at all, this condition is automatically true.)

```
In[14]:= Map[equal[V, class[x, #]] &,
           SubstTest[implies, and[subclass[u, union[v, set[x]]], member[x, v]],
            subclass[u, v], {u → P[x], v → x}]] // Reverse
```

```
Out[14]= subclass[fix[composite[inverse[SUCC], S, POWER]], RUSSELL] == True
```

```
In[15]:= subclass[fix[composite[inverse[SUCC], S, POWER]], RUSSELL] := True
```

---

invariance theorem

Lemma.

```
In[16]:= SubstTest[subclass, image[u, fix[v]], fix[composite[u, v, inverse[u]]],
           {u → eqv[x], v → composite[eqv[x], y]}] // Reverse
```

```
Out[16]= subclass[image[eqv[x], fix[composite[eqv[x], y]]],
           fix[composite[eqv[x], y, eqv[x]]]] == True
```

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

Theorem.

```
In[18]:= Map[not, SubstTest[and, implies[p0, p3], implies[p1, p2],
             implies[and[p2, p3], p4], not[implies[and[p0, p1], p4]],
             {p0 → equal[t, image[eqv[x], fix[composite[eqv[x], y]]]], p1 -> commute[y, eqv[x]],
              p2 -> equal[fix[composite[eqv[x], y, eqv[x]]], fix[composite[eqv[x], y]]],
              p3 → subclass[t, fix[composite[eqv[x], y, eqv[x]]]],
              p4 -> subclass[t, fix[composite[eqv[x], y]]]}] /.
            t -> image[eqv[x], fix[composite[eqv[x], y]]]] // Reverse
```

```
Out[18]= or[not[equal[composite[y, eqv[x]], composite[eqv[x], y]]],
           subclass[image[eqv[x], fix[composite[eqv[x], y]]], fix[composite[eqv[x], y]]] == True
```

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

The inclusion can be strengthened to an equation. A lemma is needed for this:

```
In[20]:= SubstTest[subclass, fix[composite[t, y]], range[t], t → eqv[x]] // Reverse
```

```
Out[20]= subclass[fix[composite[eqv[x], y]], fix[eqv[x]]] == True
```

```
In[21]:= subclass[fix[composite[eqv[x_], y_]], fix[eqv[x_]]] := True
```

Theorem.

```
In[22]:= SubstTest[and, implies[p, subclass[u, v]], subclass[v, u],
           {p → commute[eqv[x], y],
            u -> image[eqv[x], fix[composite[eqv[x], y]]], v -> fix[composite[eqv[x], y]]}]
```

```
Out[22]= or[equal[fix[composite[eqv[x], y]], image[eqv[x], fix[composite[eqv[x], y]]]],
           not[equal[composite[y, eqv[x]], composite[eqv[x], y]]]] == True
```

```
In[23]:= or[equal[fix[composite[eqv[x_], y_]], image[eqv[x_], fix[composite[eqv[x_], y_]]]],
           not[equal[composite[y_, eqv[x_]], composite[eqv[x_], y_]]]] := True
```

Corollary. (Restatement without wrappers.)

```
In[24]:= SubstTest[implies, and[equal[x, eqv[t]], commute[x, y]],
           equal[image[x, fix[composite[x, y]]], fix[composite[x, y]]], t → x] // Reverse

Out[24]= or[equal[fix[composite[x, y]], image[x, fix[composite[x, y]]]],
           not[equal[composite[x, y], composite[y, x]]], not[EQUIVALENCE[x]]] == True

In[25]:= or[equal[fix[composite[x_, y_]], image[x_, fix[composite[x_, y_]]]],
           not[equal[composite[x_, y_], composite[y_, x_]]], not[EQUIVALENCE[x_]]] := True
```

Application.

```
In[26]:= SubstTest[implies, and[commute[x, y], EQUIVALENCE[x]],
           equal[fix[composite[x, y]], image[x, fix[composite[x, y]]]], {x → Q, y → K}] // Reverse

Out[26]= equal[fix[composite[Q, K]], image[Q, fix[composite[Q, K]]]] == True

In[27]:= image[Q, fix[composite[Q, K]]] := fix[composite[Q, K]]
```

comments

Comment. A similar invariance theorem already occurs in the **GOEDEL** program:

```
In[28]:= image[Q, DEDEKIND]

Out[28]= DEDEKIND
```

This theorem could be derived in the same way as the above, but replacing **K** with **PS**. Note that:

```
In[29]:= fix[composite[Q, PS]]

Out[29]= complement[DEDEKIND]

In[30]:= invariant[x, complement[y]] == invariant[inverse[x], y]

Out[30]= True
```

serendipity: allclosed results

Theorem.

```
In[31]:= allclosed[union[x, y]] // Normality

Out[31]= allclosed[union[x, y]] == intersection[allclosed[x], allclosed[y]]
```

```
In[32]:= allclosed[union[x_, y_]] := intersection[allclosed[x], allclosed[y]]
```

Theorem.

```
In[33]:= allclosed[composite[id[x], inverse[SINGLETON]]] // Normality
Out[33]= allclosed[composite[id[x], inverse[SINGLETON]]] == V
In[34]:= allclosed[composite[id[x_], inverse[SINGLETON]]] := V
```

---

binclosed DIF result

Lemma.

```
In[35]:= SubstTest[implies, and[invariant[x, y], invariant[x, t]],
           invariant[x, intersection[y, t]], t → complement[z]] // Reverse
Out[35]= or[equal[0, intersection[z, image[x, intersection[y, complement[z]]]]],
           not[subclass[image[x, y], y]], not[subclass[image[inverse[x], z], z]]] == True
In[36]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[37]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[y, z],
           implies[and[member[y, u], member[z, v]], member[dif[y, z], u]],
           {u → invar[x], v → invar[inverse[x]]}]]
Out[37]= subclass[image[DIF, cart[invar[x], invar[inverse[x]]]], invar[x]] == True
In[38]:= subclass[image[DIF, cart[invar[x_], invar[inverse[x_]]]], invar[x_]] := True
```

This result, as stated, is somewhat trivial when applied to the equipollence relation **Q**. The problem is that **invar[Q]** only holds invariant subsets, and not invariant proper classes. For **Q** there are few sets that are invariant:

```
In[39]:= invar[Q]
Out[39]= succ[set[0]]
```

Corollary.

```
In[40]:= SubstTest[subclass,
           image[DIF, cart[invar[x], invar[inverse[x]]]], invar[x], x → Q] // Reverse
Out[40]= subclass[image[DIF, cart[succ[set[0]], succ[set[0]]]], succ[set[0]]] == True
In[41]:= subclass[image[DIF, cart[succ[set[0]], succ[set[0]]]], succ[set[0]]] := True
```

Restatement:

In[42]:= **member[succ[set[0]], binclosed[DIF]]**

Out[42]= True