K and PS

Johan G. F. Belinfante
2007 August 14

```
In[1]:= SetDirectory["l:"]; << goedel96.11a; << tools.m

        :Package Title: goedel96.11a           2007 August 11 at 8:20 p.m.

        It is now:  2007 Aug 14 at 23:10

        Loading Simplification Rules

        TOOLS.M                                    Revised 2007 June 25

        weightlimit = 40
```

summary

New rewrite rules involving **K** and **PS** are derived.

transitivity properties

The following variable-free rule can be interpreted using variables.

```
In[2]:= subclass[composite[K, S], S]

Out[2]= True
```

Theorem.

```
In[4]:= Map[not, SubstTest[and, implies[p2, p3],
           implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]], {p1 → subclass[z, x],
            p2 -> member[pair[x, y], K], p3 → subclass[x, y], p4 → subclass[z, y]}]] // Reverse

Out[4]= or[not[member[pair[x, y], K]], not[subclass[z, x]], subclass[z, y]] == True

In[5]:= or[not[member[pair[x_, y_], K]], not[subclass[z_, x_]], subclass[z_, y_]] := True
```

A similar result holds for this rule:

```
In[6]:= subclass[composite[S, K], S]

Out[6]= True
```

Theorem.

```
In[11]:= Map[not, SubstTest[and, implies[p2, p3], implies[and[p1, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → subclass[y, z], p2 -> member[pair[x, y], K],
            p3 → subclass[x, y], p4 → subclass[x, z]}]] // Reverse

Out[11]= or[not[member[pair[x, y], K]], not[subclass[y, z]], subclass[x, z]] == True

In[12]:= or[not[member[pair[x_, y_], K]], not[subclass[y_, z_]], subclass[x_, z_]] := True
```

fix rules

Lemma.

```
In[13]:= fix[composite[K, inverse[PS], inverse[PS]]] // Normality

Out[13]= fix[composite[K, inverse[PS], inverse[PS]]] == 0

In[14]:= fix[composite[K, inverse[PS], inverse[PS]]] := 0
```

Theorem.

```
In[15]:= SubstTest[composite, x, id[domain[x]],
          x -> intersection[composite[PS, inverse[K]], inverse[PS]]]

Out[15]= intersection[composite[PS, inverse[K]], inverse[PS]] == 0

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

Corollary.

```
In[17]:= intersection[PS, composite[K, inverse[PS]]] // DoubleInverse

Out[17]= intersection[PS, composite[K, inverse[PS]]] == 0

In[18]:= intersection[PS, composite[K, inverse[PS]]] := 0
```

Corollary.

```
In[19]:= fix[composite[PS, inverse[K], PS]] // Renormality

Out[19]= fix[composite[PS, inverse[K], PS]] == 0

In[20]:= fix[composite[PS, inverse[K], PS]] := 0
```

intersections

Lemma.

```
In[51]:= member[0, fix[composite[PS, inverse[K], S]]] // AssertTest

Out[51]= member[0, fix[composite[PS, inverse[K], S]]] == False
```

```
In[52]:= % /. Equal → SetDelayed
```

Lemma.

```
In[62]:= SubstTest[implies, subclass[u, v], subclass[fix[u], fix[v]],
           {u → composite[PS, inverse[K]], v → composite[PS, inverse[K], S]}] // Reverse

Out[62]= equal[V, union[fix[composite[PS, inverse[K], S]], set[0]]] == True

In[63]:= % /. Equal → SetDelayed
```

Theorem.

```
In[65]:= equal[fix[composite[PS, inverse[K], S]], complement[set[0]]] // AssertTest

Out[65]= equal[complement[set[0]], fix[composite[PS, inverse[K], S]]] == True

In[67]:= fix[composite[PS, inverse[K], S]] := complement[set[0]]
```

Corollary.

```
In[68]:= equal[id[complement[set[0]]], intersection[S, composite[K, inverse[PS]]]] // AssertTest

Out[68]= equal[id[complement[set[0]]], intersection[S, composite[K, inverse[PS]]]] == True

In[70]:= intersection[S, composite[K, inverse[PS]]] := id[complement[set[0]]]
```

Corollary.

```
In[73]:= intersection[composite[PS, inverse[K]], inverse[S]] // DoubleInverse

Out[73]= intersection[composite[PS, inverse[K]], inverse[S]] == id[complement[set[0]]]

In[74]:= intersection[composite[PS, inverse[K]], inverse[S]] := id[complement[set[0]]]
```

---

a special result

Theorem.

```
In[76]:= Map[not, SubstTest[and, implies[and[p1, p2, p3, p4, p5], p6], implies[and[p4, p6], p7],
            not[implies[and[p1, p2, p3, p4, p5], p7]], {p1 → subclass[u, v],
             p2 → not[equal[u, v]], p3 → member[pair[u, x], K], p4 -> member[pair[v, y], K],
             p5 → subclass[v, x], p6 → equal[v, x], p7 → subclass[x, y]}]] // Reverse

Out[76]= or[equal[u, v], not[member[pair[u, x], K]], not[member[pair[v, y], K]],
           not[subclass[u, v]], not[subclass[v, x]], subclass[x, y]] == True

In[77]:= or[equal[u_, v_], not[member[pair[u_, x_], K]], not[member[pair[v_, y_], K]],
           not[subclass[u_, v_]], not[subclass[v_, x_]], subclass[x_, y_]] := True
```

comparable and covering lemma

Temporary definition.

```
In[78]:= comparable[x_, y_] := or[subclass[x, y], subclass[y, x]]
```

The rewrite rules derived in this notebook enable the **GOEDEL** program to recognize the following pretty lemma which occurred in a handout containing a hand-produced proof of Zorn's lemma from the axiom of choice that the author wrote in July 1972 for an abstract algebra class that he taught at Carnegie-Mellon University.

```
In[82]:= implies[and[member[pair[u, x], K], member[pair[v, y], K], comparable[u, v],
             comparable[u, y], comparable[v, x]], or[equal[u, v], comparable[x, y]]] // not // not

Out[82]= True
```