ids[CATOFUNS]

Johan G. F. Belinfante
2008 December 23

```
In[1]:= SetDirectory["l:"]; << goedel.08dec23a;<< tools.m

        :Package Title: goedel.08dec23a         2008 December 23 at 2:20 p.m.

        It is now:  2008 Dec 23 at 22:30

        Loading Simplification Rules

        TOOLS.M                                      Revised 2008 December 16

        weightlimit = 40
```

summary

In this notebook **APPLY** rules are derived for **CATOFUNS**, and used to show that the class **ids[CATOFUNS]** of identity morphisms for the category of sets is equal to the class **inverse[IMAGE[DUP]]** of ordered pairs of the form **pair[id[x], x]**, where **x** is a set.

A-cart lemmas

Lemma.

```
In[2]:= A[cart[intersection[x, set[y]], z]] // Renormality

Out[2]= A[cart[intersection[x, set[y]], z]] ==
         union[A[cart[set[y], z]], complement[image[V, intersection[x, set[y]]]]]

In[3]:= A[cart[intersection[x_, set[y_]], z_]] :=
         union[A[cart[set[y], z]], complement[image[V, intersection[x, set[y]]]]]
```

Lemma.

```
In[4]:= A[cart[x, intersection[y, set[z]]]] // Renormality

Out[4]= A[cart[x, intersection[y, set[z]]]] ==
         union[A[cart[x, set[z]]], complement[image[V, intersection[y, set[z]]]]]

In[5]:= A[cart[x_, intersection[y_, set[z_]]]] :=
         union[A[cart[x, set[z]]], complement[image[V, intersection[y, set[z]]]]]
```

vertical section and APPLY rules

```
In[6]:= simplify = False; cond = False;
```

Theorem. (This takes quite a while.)

```
In[7]:= ImageComp[direct[COMPOSE, FIRST], id[domain[CATOFUNS]],
         cart[cart[set[x], set[u]], cart[set[y], set[v]]]]

Out[7]= image[CATOFUNS, cart[cart[set[x], set[u]], cart[set[y], set[v]]]] ==
         cart[set[composite[x, y]],
          intersection[complement[image[V, intersection[complement[v], range[y]]]], image[V,
            intersection[FUNS, set[y]]], image[V, intersection[map[v, u], set[x]]], set[u]]]

In[8]:= image[CATOFUNS, cart[cart[set[x_], set[u_]], cart[set[y_], set[v_]]]] :=
         cart[set[composite[x, y]],
          intersection[complement[image[V, intersection[complement[v], range[y]]]], image[V,
            intersection[FUNS, set[y]]], image[V, intersection[map[v, u], set[x]]], set[u]]]
```

Theorem.

```
In[9]:= Map[A, SubstTest[image, funpart[z],
          set[w], {w → PAIR[PAIR[x, u], PAIR[y, v]], z → CATOFUNS}]]

Out[9]= APPLY[CATOFUNS, PAIR[PAIR[x, u], PAIR[y, v]]] ==
         union[complement[image[V, intersection[FUNS, set[y]]]],
          complement[image[V, intersection[map[v, u], set[x]]]],
          image[V, intersection[complement[v], range[y]]], PAIR[composite[x, y], u]]

In[10]:= APPLY[CATOFUNS, PAIR[PAIR[x_, u_], PAIR[y_, v_]]] :=
          union[complement[image[V, intersection[FUNS, set[y]]]],
           complement[image[V, intersection[map[v, u], set[x]]]],
           image[V, intersection[complement[v], range[y]]], PAIR[composite[x, y], u]]
```

APPLY and PAIR

Theorem.

```
In[11]:= Assoc[direct[COMPOSE, FIRST],
           id[domain[CATOFUNS]], id[cart[cart[V, V], cart[V, V]]]] // Reverse

Out[11]= composite[CATOFUNS, id[cart[cart[V, V], cart[V, V]]]] == CATOFUNS

In[12]:= composite[CATOFUNS, id[cart[cart[V, V], cart[V, V]]]] := CATOFUNS
```

Theorem.

```
In[13]:= equal[union[complement[image[V, set[first[x]]]], PAIR[x, y]],
           PAIR[PAIR[first[x], second[x]], y]]

Out[13]= True

In[14]:= union[complement[image[V, set[first[x_]]]], PAIR[x_, y_]] :=
           PAIR[PAIR[first[x], second[x]], y]

In[15]:= SubstTest[APPLY, SWAP, union[complement[image[V, set[w]]], PAIR[y, x]], w -> first[y]]

Out[15]= union[complement[image[V, set[first[y]]]], PAIR[x, y]] ==
           PAIR[x, PAIR[first[y], second[y]]]

In[16]:= union[complement[image[V, set[first[y_]]]], PAIR[x_, y_]] :=
           PAIR[x, PAIR[first[y], second[y]]]
```

a second APPLY rule

```
In[17]:= Map[A, ImageComp[CATOFUNS, id[cart[cart[V, V], cart[V, V]]], set[PAIR[x, y]]]]

Out[17]= APPLY[CATOFUNS, PAIR[x, y]] ==
           union[complement[image[V, intersection[FUNS, set[first[y]]]]],
            complement[image[V, intersection[map[second[y], second[x]], set[first[x]]]]],
            image[V, intersection[complement[second[y]], range[first[y]]]],
            PAIR[composite[first[x], first[y]], second[x]]]

In[18]:= APPLY[CATOFUNS, PAIR[x_, y_]] :=
           union[complement[image[V, intersection[FUNS, set[first[y]]]]],
            complement[image[V, intersection[map[second[y], second[x]], set[first[x]]]]],
            image[V, intersection[complement[second[y]], range[first[y]]]],
            PAIR[composite[first[x], first[y]], second[x]]]
```

simplification rules

```
In[19]:= simplify = True; cond = True;
```

Lemma.

```
In[20]:= SubstTest[and, equal[v, domain[x]], FUNCTION[x],
            member[x, V], subclass[range[x], u], v → domain[x]] // Reverse

Out[20]= and[FUNCTION[x], member[x, V], subclass[range[x], u]] == member[x, map[domain[x], u]]

In[21]:= and[FUNCTION[x_], member[x_, V], subclass[range[x_], u_]] :=
           member[x, map[domain[x], u]]
```

Lemma.

```
In[22]:= equiv[and[member[y, V], member[x, map[y, z]]], member[x, map[y, z]]]

Out[22]= True

In[23]:= and[member[x_, map[y_, z_]], member[y_, V]] := member[x, map[y, z]]
```

second vertical section rule

```
In[24]:= simplify = False; cond = False;

In[25]:= SubstTest[image, funpart[w], set[PAIR[x, y]], w → CATOFUNS] // Reverse

Out[25]= image[CATOFUNS, cart[set[x], set[y]]] ==
          cart[set[composite[first[x], first[y]]], intersection[
            complement[image[V, intersection[complement[second[y]], range[first[y]]]]],
            image[V, intersection[FUNS, set[first[y]]]],
            image[V, intersection[map[second[y], second[x]], set[first[x]]], set[second[x]]]]

In[26]:= image[CATOFUNS, cart[set[x_], set[y_]]] :=
          cart[set[composite[first[x], first[y]]], intersection[
            complement[image[V, intersection[complement[second[y]], range[first[y]]]]],
            image[V, intersection[FUNS, set[first[y]]]],
            image[V, intersection[map[second[y], second[x]], set[first[x]]], set[second[x]]]]
```

ids[CATOFUNS]

```
In[27]:= simplify = True; cond = True;
```

Lemma

```
In[28]:= SubstTest[implies, member[pair[s, t], funpart[r]],
            equal[APPLY[funpart[r], s], t], {r → CATOFUNS,
             s → pair[pair[id[range[y]], range[y]], pair[y, v]], t → pair[y, v]}] // Reverse

Out[28]= or[equal[APPLY[CATOFUNS, pair[pair[id[range[y]], range[y]], pair[y, v]]], pair[y, v]],
            not[equal[v, range[y]]], not[FUNCTION[y]],
            not[member[v, V]], not[member[y, V]]] == True

In[29]:= (% /. {v → v_, y → y_}) /. Equal → SetDelayed

In[30]:= SubstTest[implies, member[pair[s, t], funpart[r]], equal[APPLY[funpart[r], s], t],
            {r → CATOFUNS, s → pair[pair[id[v], v], pair[y, v]], t → pair[y, v]}] // Reverse

Out[30]= or[equal[APPLY[CATOFUNS, pair[pair[id[v], v], pair[y, v]]], pair[y, v]],
            not[member[v, V]], not[member[y, map[domain[y], v]]]] == True

In[31]:= (% /. {v → v_, y → y_}) /. Equal → SetDelayed
```

```
In[32]:= SubstTest[or, equal[y, APPLY[x, pair[y, z]]], not[member[y, V]], not[member[z, V]],
           not[member[pair[y, z], domain[x]]], not[subclass[composite[x, RIGHT[z]], Id]],
           {x → CATOFUNS, y -> pair[id[v], v], z -> pair[t, v]}] // Reverse

Out[32]= or[equal[APPLY[CATOFUNS, pair[pair[id[v], v], pair[t, v]]], pair[id[v], v]],
           not[member[t, map[domain[t], v]]], not[member[v, V]],
           not[subclass[composite[CATOFUNS, RIGHT[pair[t, v]]], Id]]] == True

In[33]:= (% /. {t → t_, y → y_}) /. Equal → SetDelayed

In[34]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[p2, p4], not[implies[and[p1, p2], p5]],
            {p1 -> subclass[composite[CATOFUNS, RIGHT[pair[y, v]]], Id],
             p2 → member[pair[y, v], range[CATOFUNS]],
             p3 → equal[APPLY[CATOFUNS, pair[pair[id[v], v], pair[y, v]]], pair[id[v], v]],
             p4 -> equal[APPLY[CATOFUNS, pair[pair[id[v], v], pair[y, v]]], pair[y, v]],
             p5 → equal[id[v], y]}]] // Reverse

Out[34]= or[equal[y, id[v]], not[member[v, V]], not[member[y, map[domain[y], v]]],
           not[subclass[composite[CATOFUNS, RIGHT[pair[y, v]]], Id]]] == True

In[35]:= (% /. {v → v_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[36]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[y, v],
           implies[and[member[pair[y, v], r], subclass[composite[s, RIGHT[pair[y, v]]], Id]],
            member[pair[y, v], t]], {r → range[CATOFUNS], s → CATOFUNS, t → inverse[IDP]}]]

Out[36]= subclass[composite[id[FUNS], intersection[
             complement[inverse[range[fix[composite[inverse[FIRST], Di, CATOFUNS]]]]],
             composite[inverse[IMAGE[SECOND]], inverse[S]]]], IMAGE[DUP]] == True

In[37]:= % /. Equal → SetDelayed
```

Theorem.

```
In[38]:= SubstTest[implies, and[member[x, y], subclass[y, z]],
           member[x, z], {y → inverse[composite[id[FUNS], intersection[complement[
                 inverse[range[fix[composite[inverse[FIRST], Di, CATOFUNS]]]]], composite[
                 inverse[IMAGE[SECOND]], inverse[S]]]]], z → inverse[IMAGE[DUP]]}] // Reverse

Out[38]= or[equal[first[x], id[second[x]]],
           not[member[first[x], map[domain[first[x]], second[x]]]],
           not[subclass[composite[CATOFUNS, RIGHT[x]], Id]]] == True

In[39]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[40]:= Map[or[subclass[ids[CATOFUNS], cart[FUNS, V]], #] &,
          SubstTest[subclass, ids[t], range[t], t → CATOFUNS] // Reverse]

Out[40]= subclass[ids[CATOFUNS], cart[FUNS, V]] == True
```

```
In[41]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[42]:= SubstTest[subclass, ids[t], range[t], t→ CATOFUNS] // Reverse

Out[42]= subclass[ids[CATOFUNS], composite[S, IMAGE[SECOND]]] == True

In[43]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[44]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
           member[u, w], {u→ x, v -> ids[CATOFUNS], w -> range[CATOFUNS]}] // Reverse

Out[44]= or[member[first[x], map[domain[first[x]], second[x]]],
           not[member[x, ids[CATOFUNS]]]] == True

In[45]:= (% /. x→ x_) /. Equal→ SetDelayed
```

Theorem.

```
In[46]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
            implies[and[p2, p3], p4], not[implies[p1, p4]], {p1→ member[x, ids[CATOFUNS]],
             p2→ member[x, range[CATOFUNS]], p3 -> subclass[composite[CATOFUNS, RIGHT[x]], Id],
             p4→ member[x, inverse[IDP]]}]] // Reverse

Out[46]= or[equal[first[x], id[second[x]]], not[member[x, ids[CATOFUNS]]]] == True

In[47]:= (% /. x→ x_) /. Equal→ SetDelayed
```

Theorem.

```
In[48]:= Map[equal[V, #] &, SubstTest[class, x,
           implies[member[x, u], member[x, v]], {u→ ids[CATOFUNS], v→ inverse[IDP]}]]

Out[48]= subclass[ids[CATOFUNS], inverse[IMAGE[DUP]]] == True

In[49]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[50]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u -> composite[CATOFUNS, LEFT[pair[id[x], x]]],
            v -> composite[CATORELN, LEFT[pair[id[x], x]]], w→ Id}] // Reverse

Out[50]= subclass[composite[CATOFUNS, LEFT[pair[id[x], x]]], Id] == True

In[51]:= (% /. x→ x_) /. Equal→ SetDelayed
```

Lemma.

```
In[52]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u -> composite[CATOFUNS, RIGHT[pair[id[x], x]]],
            v -> composite[CATORELN, RIGHT[pair[id[x], x]]], w → Id}] // Reverse

Out[52]= subclass[composite[CATOFUNS, RIGHT[pair[id[x], x]]], Id] == True

In[53]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem.

```
In[54]:= member[pair[id[x], x], ids[CATOFUNS]] // AssertTest

Out[54]= member[pair[id[x], x], ids[CATOFUNS]] == member[x, V]

In[55]:= member[pair[id[x_], x_], ids[CATOFUNS]] := member[x, V]
```

Theorem.

```
In[56]:= fix[composite[ids[CATOFUNS], IMAGE[DUP]]] // Normality

Out[56]= fix[composite[ids[CATOFUNS], IMAGE[DUP]]] == V

In[57]:= % /. Equal→ SetDelayed

In[58]:= Map[subclass[#, ids[CATOFUNS]] &,
           ImageComp[inverse[composite[SECOND, id[inverse[IMAGE[DUP]]]]],
            composite[SECOND, id[inverse[IMAGE[DUP]]]], ids[CATOFUNS]]] // Reverse

Out[58]= subclass[inverse[IMAGE[DUP]], ids[CATOFUNS]] == True

In[59]:= % /. Equal→ SetDelayed

In[60]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> inverse[IMAGE[DUP]], v -> ids[CATOFUNS]}]

Out[60]= equal[ids[CATOFUNS], inverse[IMAGE[DUP]]] == True

In[61]:= ids[CATOFUNS] := inverse[IMAGE[DUP]]
```