one-sided inverses in the category of sets

Johan G. F. Belinfante
2009 September 25

```
In[1]:= SetDirectory["l:"]; << goedel.09sep23a;<< tools.m

:Package Title: goedel.09sep23a              2009 September 23 at 11:35 a.m.

It is now:  2009 Sep 25 at 10:14

Loading Simplification Rules

TOOLS.M                                     Revised 2009 September 15

weightlimit = 40
```

summary

This notebook is only concerned with one-sided invertibity in the category of sets. Morphisms in the category of sets are ordered pairs **pair[u, y]** such that **u ∈ map[x, y]** where **x = domain[u]**. When the product of two morphisms in a category is an identity morphism, the left-hand factor is said to be **right-invertible** or a **retraction** and the right-hand factor is said to be **left-invertible** or a **section**. A membership rule is derived in this notebook for the class **image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]** of ordered pairs **pair[pair[u, x], pair[v, y]]** of morphisms whose product is an identity morphism. The domain of this class is the class of retractions, and its range is the class of sections. A morphism **pair[u, x]** in the category of sets is said to be **onto** if **x = range[u]**. A morphism **pair[v, y]** in the category of sets is said to be **one-to-one** if **inverse[v]** is a function. It is shown that retractions in the category of sets are onto, and that sections are one-to-one. A counterexample for a converse statement is presented: the one-to-one morphism **PAIR[0, set[0]]** is not a section in the category of sets.

a COMPOSE rule

The composition law **CATOFUNS** for the category of sets is a certain restriction of the direct product of the two associative functions **COMPOSE** and **FIRST**. A special rewrite rule is needed in this section for the class **inverse[IMAGE[DUP]] ∘ COMPOSE** of ordered triples **pair[pair[x, y], z]** such that **x ∘ y = id[z]**.

Lemma.

```
In[2]:= SubstTest[member, pair[w, z], composite[inverse[funpart[u]], funpart[v]],
          {u → IMAGE[DUP], v → COMPOSE, w → pair[x, y]}] // Reverse

Out[2]= member[pair[pair[x, y], z], composite[inverse[IMAGE[DUP]], COMPOSE]] ==
         and[equal[composite[x, y], id[z]], member[x, V], member[y, V], member[z, V]]
```

```
In[3]:= member[pair[pair[x_, y_], z_], composite[inverse[IMAGE[DUP]], COMPOSE]] :=
          and[equal[composite[x, y], id[z]], member[x, V], member[y, V], member[z, V]]
```

Lemma.

```
In[4]:= Map[implies[member[pair[pair[u, x], pair[v, y]], #], equal[composite[u, v], id[x]]] &,
          IminComp[direct[COMPOSE, FIRST], id[domain[CATOFUNS]], inverse[IDP]]]

Out[4]= or[equal[composite[u, v], id[x]], not[member[pair[pair[u, x], pair[v, y]],
            image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]]] == True

In[5]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= Map[implies[#, equal[x, domain[v]]] &, Map[member[pair[pair[u, x], pair[v, y]], #] &,
           IminComp[direct[COMPOSE, FIRST], id[domain[CATOFUNS]], inverse[IDP]]]]

Out[6]= or[equal[x, domain[v]], not[member[pair[pair[u, x], pair[v, y]],
            image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]]] == True

In[7]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[8]:= SubstTest[and, implies[p, q], implies[p, r], {p ->
             member[pair[pair[u, x], pair[v, y]], image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]],
           q -> member[v, map[domain[v], y]], r → equal[x, domain[v]]}]

Out[8]= or[member[v, map[x, y]], not[member[pair[pair[u, x], pair[v, y]],
            image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]]] == True

In[9]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. Reverse implication.

```
In[10]:= Map[implies[
              and[member[u, map[y, x]], member[v, map[x, y]], equal[composite[u, v], id[x]]], #] &,
            Map[member[pair[pair[u, x], pair[v, y]], #] &,
             IminComp[direct[COMPOSE, FIRST], id[domain[CATOFUNS]], inverse[IDP]]]] // MapNotNot

Out[10]= or[member[pair[pair[u, x], pair[v, y]], image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]],
            not[equal[composite[u, v], id[x]]],
            not[member[u, map[y, x]]], not[member[v, map[x, y]]]] == True

In[11]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

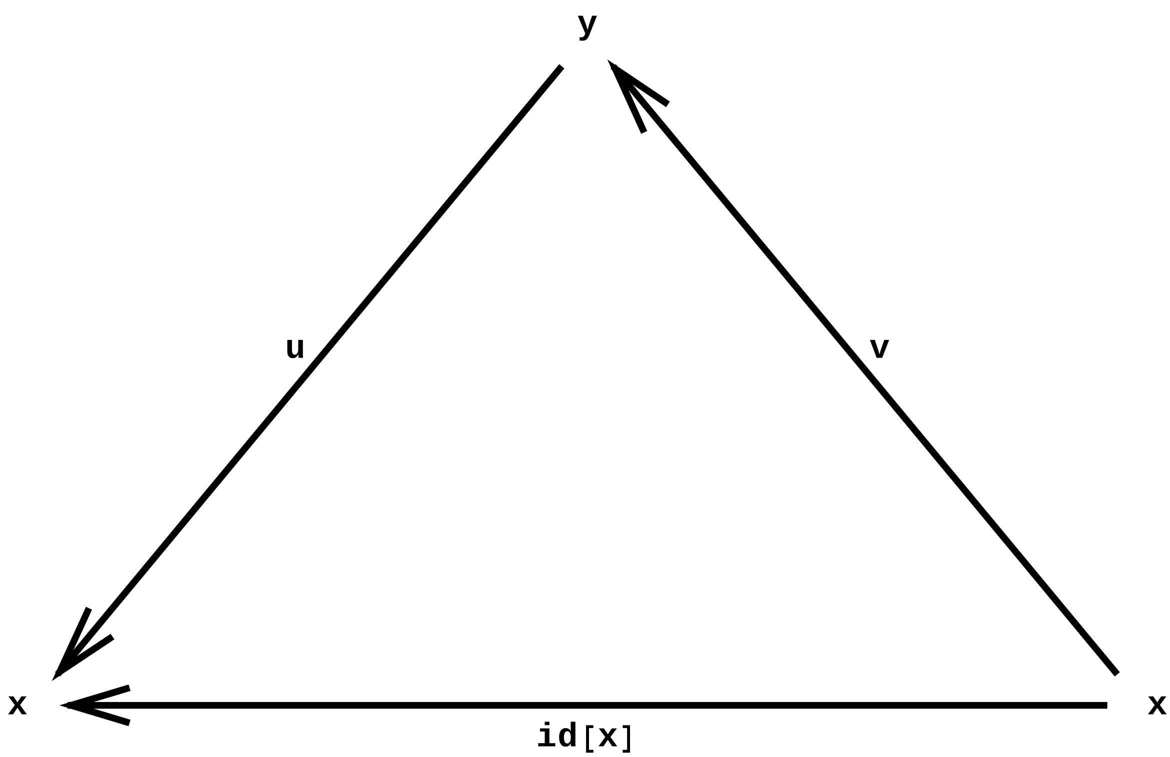
Theorem. Membership rule for the one-sided-inverse relation in the category of sets.

```
In[12]:= equiv[member[pair[pair[u, x], pair[v, y]],
              image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]], and[member[u, map[y, x]],
              member[v, map[x, y]], equal[composite[u, v], id[x]]]] // not // not

Out[12]= True
```

```
In[13]:= member[pair[pair[u_, x_], pair[v_, y_]],
           image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]] :=
          and[equal[composite[u, v], id[x]], member[u, map[y, x]], member[v, map[x, y]]]
```

The following diagram provides a picture for this membership rule.



retractions are onto

Theorem. Retractions in the category of sets are onto.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[and[p1, p2], p3], not[implies[p1, p3]], {p1 -> member[
               pair[pair[u, x], pair[v, y]], image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]],
             p2 → subclass[composite[u, v], Id], p3 → equal[range[u], x]}]] // Reverse

Out[14]= or[equal[x, range[u]], not[equal[composite[u, v], id[x]]],
           not[member[u, map[y, x]]], not[member[v, map[x, y]]]] == True

In[15]:= or[equal[x_, range[u_]], not[equal[composite[u_, v_], id[x_]]],
           not[member[u_, map[y_, x_]]], not[member[v_, map[x_, y_]]]] := True
```

A variable-free reformulation of this result will now be derived.

Lemma.

```
In[16]:= Map[composite[id[cart[V, V]], complement[#], id[cart[V, V]]] &,
          SubstTest[class, pair[pair[u, x], pair[v, y]],
           implies[member[pair[pair[u, x], pair[v, y]], t], equal[x, range[u]]],
           t -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]]

Out[16]= composite[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]],
           id[composite[Id, complement[IMAGE[SECOND]]]]] == 0

In[17]:= % /. Equal → SetDelayed
```

Theorem.

```
In[18]:= SubstTest[empty, composite[id[t], dif[u, v], id[t]],
          {t → cart[V, V], u -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]],
           v → cart[IMAGE[SECOND], cart[V, V]]}]

Out[18]= subclass[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]],
           cart[IMAGE[SECOND], cart[V, V]]] == True

In[19]:= subclass[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]],
           cart[IMAGE[SECOND], cart[V, V]]] := True
```

Corollary.

```
In[20]:= SubstTest[implies, subclass[u, v], subclass[domain[u], domain[v]],
           {u -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]],
            v -> cart[IMAGE[SECOND], cart[V, V]]}] // Reverse

Out[20]= subclass[domain[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]], IMAGE[SECOND]] == True

In[21]:= subclass[domain[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]], IMAGE[SECOND]] := True
```

Corollary.

```
In[22]:= SubstTest[and, subclass[u, cart[V, V]], subclass[domain[u], v],
          {u -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], v -> IMAGE[SECOND]}]

Out[22]= subclass[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], cart[IMAGE[SECOND], V]] == True

In[23]:= subclass[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], cart[IMAGE[SECOND], V]] := True
```

sections are one-to-one

Sections in the category of sets are one-to-one. A slightly more general result can be derived.

Theorem. If the product of two morphisms in the category of sets is an inclusion morphism, then the right-hand factor is one-to-one.

```
In[24]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> and[equal[composite[u, v], id[w]], member[u, map[y, z]], member[v, map[x, y]]],
            p2 -> subclass[composite[u, v], Id], p3 -> FUNCTION[inverse[v]]}]] // Reverse

Out[24]= or[FUNCTION[inverse[v]], not[equal[composite[u, v], id[w]]],
          not[member[u, map[y, z]]], not[member[v, map[x, y]]]] == True

In[25]:= or[FUNCTION[inverse[v_]], not[equal[composite[u_, v_], id[w_]]],
          not[member[u_, map[y_, z_]]], not[member[v_, map[x_, y_]]]] := True
```

A variable-free formulation of the fact that sections are one-to-one will now be derived.

Lemma.

```
In[26]:= implies[member[pair[pair[u, x], pair[v, y]],
           image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]], member[v, BIJ]] // NotNotTest

Out[26]= or[and[FUNCTION[v], FUNCTION[inverse[v]]], not[equal[composite[u, v], id[x]]],
          not[member[u, map[y, x]]], not[member[v, map[x, y]]]] == True

In[27]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. A variable-free statement that sections in the category of sets are one-to-one.

```
In[28]:= Map[empty[composite[id[cart[V, V]], complement[#], id[cart[V, V]]]] &,
          SubstTest[class, pair[pair[u, x], pair[v, y]],
           implies[member[pair[pair[u, x], pair[v, y]], s], member[v, t]],
           {s -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], t → BIJ}]]

Out[28]= subclass[domain[range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]], BIJ] == True

In[29]:= subclass[domain[range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]], BIJ] := True
```

Corollary. An upper bound for the class of sections.

```
In[30]:= SubstTest[and, subclass[u, cart[V, V]], subclass[domain[u], v],
          {u -> range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]], v → BIJ}]

Out[30]= subclass[range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]], cart[BIJ, V]] == True

In[31]:= subclass[range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]], cart[BIJ, V]] := True
```

Corollary. An upper bound for the one-sided invertibility relation.

```
In[32]:= SubstTest[and, subclass[u, cart[V, V]], subclass[range[u], v],
          {u -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], v → cart[BIJ, V]}]

Out[32]= subclass[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], cart[V, cart[BIJ, V]]] == True

In[33]:= subclass[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], cart[V, cart[BIJ, V]]] := True
```

This result may be combined with that of the preceding section.

Corollary. An upper bound for the one-sided invertibility relation.

```
In[34]:= SubstTest[subclass, t, intersection[u, v, w],
           {t -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], u → domain[CATOFUNS],
            v → cart[IMAGE[SECOND], V], w → cart[V, cart[BIJ, V]]}] // Reverse

Out[34]= subclass[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], cart[
            composite[IMAGE[SECOND], id[FUNS]], composite[S, IMAGE[SECOND], id[BIJ]]]] == True

In[35]:= subclass[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], cart[
            composite[IMAGE[SECOND], id[FUNS]], composite[S, IMAGE[SECOND], id[BIJ]]]] := True
```

counterexample

It was shown above that sections in the category of sets are one-to-one. The converse does not hold. In this section a counterexample is presented that shows that a one-to-one morphism in the category of sets need not be a section.

Lemma. A simplification rule.

```
In[83]:= equal[composite[Id, image[inverse[image[inverse[CATOFUNS], x]], y]],
          image[inverse[image[inverse[CATOFUNS], x]], y]]

Out[83]= True

In[85]:= composite[Id, image[inverse[image[inverse[CATOFUNS], x_]], y_]] :=
          image[inverse[image[inverse[CATOFUNS], x]], y]
```

Lemma. The counterexample.

```
In[88]:= Map[not[empty[#]] &,
          SubstTest[class, pair[u, x], member[pair[pair[u, x], pair[v, y]], t],
           {t -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]], v → 0, y → set[0]}]]

Out[88]= member[pair[0, set[0]], range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]] == False

In[89]:= member[pair[0, set[0]], range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]] := False
```

Theorem. A one-to-one morphism in the category of sets need not be a section.

```
In[91]:= Map[not, SubstTest[implies, and[member[r, s], subclass[s, t]],
            member[r, t], {r → pair[0, set[0]], s -> composite[S, IMAGE[SECOND], id[BIJ]],
             t → range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]}]] // Reverse

Out[91]= subclass[composite[S, IMAGE[SECOND], id[BIJ]],
           range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]] == False

In[92]:= subclass[composite[S, IMAGE[SECOND], id[BIJ]],
           range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]] := False
```