two-sided inverses in the category of sets

Johan G. F. Belinfante
2009 October 3

```
In[1]:= SetDirectory["l:"]; << goedel.09oct03a; << tools.m

        :Package Title: goedel.09oct03a             2009 October 3 at 8:50 a.m.

        It is now:  2009 Oct 3 at 8:53

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 September 15

        weightlimit = 40
```

summary

This notebook is concerned with two-sided inverses in the category of sets. A pair of morphisms in a category **cat[x]** is an **inverse pair** if the product of these two morphisms in either order is an identity morphism. The class **inv[cat[x]]** of all such invertible pairs is an involution, that is, a function which is its own inverse. For the category of sets an explicit formula for this class **inv[CATOFUNS]** can be derived.

isomorphisms

A morphism in a category **cat[x]** is **invertible**, also called an **isomorphism**, if it belongs to **domain[inv[cat[x]]]**. In this section it will be shown that the class of isomorphisms in the category of sets is equal to the class of one-to-one and onto morphisms.

The class of isomorphisms for any category is the intersection of the class of retractions (= right-invertible morphisms) and the class of sections (= left-invertible morphisms). For the category of sets, this fact is expressed by the following rewrite rule.

Theorem. A morphism in the category of sets is invertible if it is both a retraction and a section.

```
In[2]:= SubstTest[intersection, domain[image[inverse[cat[x]], ids[cat[x]]]],
          range[image[inverse[cat[x]], ids[cat[x]]]], x → CATOFUNS] // Reverse

Out[2]= intersection[domain[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]],
          range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]] == domain[inv[CATOFUNS]]

In[3]:= % /. Equal→ SetDelayed
```

Corollary. Isomorphisms are retractions.

```
In[4]:= SubstTest[subclass, intersection[u, v], u,
          {u -> domain[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]],
           v -> range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]}] // Reverse

Out[4]= subclass[domain[inv[CATOFUNS]],
          domain[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]] == True

In[5]:= % /. Equal→ SetDelayed
```

Corollary. Isomorphisms are sections.

```
In[6]:= SubstTest[subclass, intersection[u, v], v,
          {u -> domain[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]],
           v -> range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]}] // Reverse

Out[6]= subclass[domain[inv[CATOFUNS]],
          range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]] == True

In[7]:= % /. Equal→ SetDelayed
```

Since retractions (= right-invertible morphisms) in the category of sets are onto, the following corollary is immediate.

Lemma. Isomorphisms in the category of sets are onto.

```
In[8]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
          subclass[u, w], {u -> domain[inv[CATOFUNS]],
           v -> domain[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]],
           w -> IMAGE[SECOND]}] // Reverse

Out[8]= subclass[domain[inv[CATOFUNS]], IMAGE[SECOND]] == True

In[9]:= % /. Equal→ SetDelayed
```

Since sections (= left-invertible morphisms) in the category of sets are one-to-one, the following corollary is obtained.

Lemma. Isomorphisms in the category of sets are one-to-one.

```
In[10]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u -> domain[inv[CATOFUNS]], v →
             range[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]], w → cart[BIJ, V]}] // Reverse

Out[10]= subclass[domain[inv[CATOFUNS]], cart[BIJ, V]] == True

In[11]:= % /. Equal→ SetDelayed
```

Lemma. This takes a while.

```
In[12]:= implies[member[pair[u, x], composite[IMAGE[SECOND], id[BIJ]]],
           member[pair[pair[u, x], pair[inverse[u], domain[u]]], inv[CATOFUNS]]] // AssertTest

Out[12]= or[member[pair[pair[u, x], pair[inverse[u], domain[u]]], inv[CATOFUNS]],
           not[member[u, bij[domain[u], x]]], not[member[x, V]]] == True
```

```
In[13]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

Theorem.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 → member[pair[u, x], composite[IMAGE[SECOND], id[BIJ]]],
             p2 -> member[pair[pair[u, x], pair[inverse[u], domain[u]]], inv[CATOFUNS]],
             p3 -> member[pair[u, x], domain[inv[CATOFUNS]]]}]] // Reverse

Out[14]= or[member[pair[u, x], domain[inv[CATOFUNS]]],
           not[member[u, bij[domain[u], x]]], not[member[x, V]]] == True

In[15]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

Lemma. (Elimination of the variables.)

```
In[16]:= Map[composite[Id, complement[#]] &,
          SubstTest[class, pair[u, x], not[member[pair[u, x], t]],
           t -> dif[composite[IMAGE[SECOND], id[BIJ]], domain[inv[CATOFUNS]]]]]

Out[16]= composite[intersection[complement[domain[inv[CATOFUNS]]], IMAGE[SECOND]], id[BIJ]] == 0

In[17]:= % /. Equal → SetDelayed
```

Lemma.

```
In[18]:= SubstTest[empty, dif[u, v],
           {u -> composite[IMAGE[SECOND], id[BIJ]], v -> domain[inv[CATOFUNS]]}]

Out[18]= subclass[composite[IMAGE[SECOND], id[BIJ]], domain[inv[CATOFUNS]]] == True

In[19]:= % /. Equal → SetDelayed
```

Theorem. The class of isomorphisms in the category of sets is the class of one-to-one and onto morphisms.

```
In[20]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> composite[IMAGE[SECOND], id[BIJ]], v -> domain[inv[CATOFUNS]]}]

Out[20]= equal[composite[IMAGE[SECOND], id[BIJ]], domain[inv[CATOFUNS]]] == True

In[21]:= domain[inv[CATOFUNS]] := composite[IMAGE[SECOND], id[BIJ]]
```

Remark. This is a temporary rewrite rule. It becomes unnecessary once an explicit formula for **inv[CATOFUNS]** has been derived.

---

a temporary membership rule

Because **CATOFUNS** is a category, the inverse-pair relation **inv[CATOFUNS]** is a function.

Theorem. A formula for the function **inv[CATOFUNS]**.

```
In[22]:= SubstTest[intersection, image[inverse[t], ids[t]],
           inverse[image[inverse[t], ids[t]]], t → CATOFUNS] // Reverse

Out[22]= intersection[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]],
           inverse[image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]]] == inv[CATOFUNS]

In[23]:= % /. Equal → SetDelayed
```

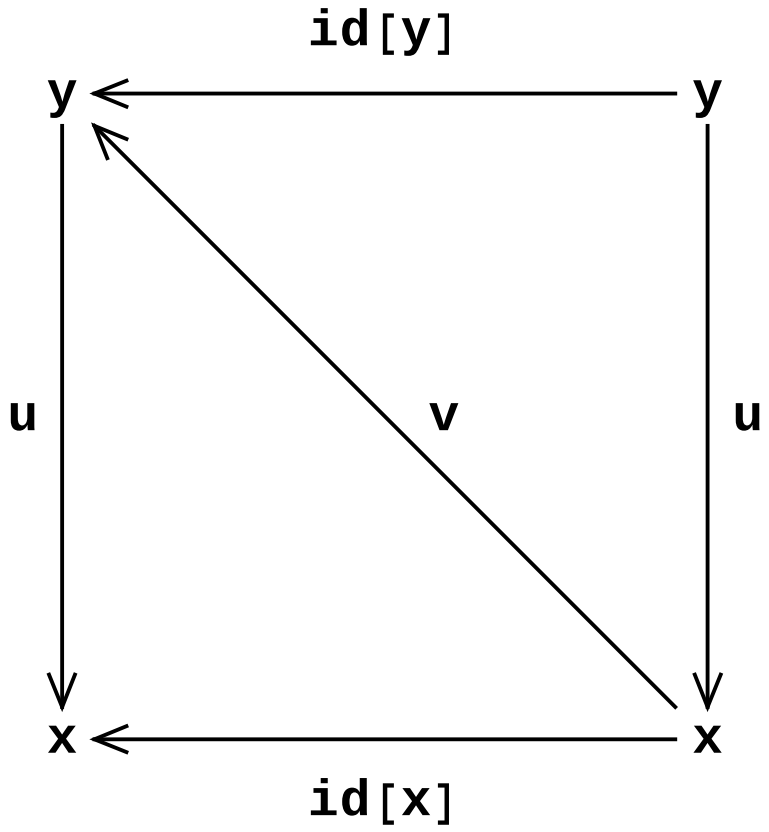
Theorem. Temporary membership rule for **inv[CATOFUNS]**.

```
In[24]:= SubstTest[member, w, intersection[t, inverse[t]], {w -> pair[pair[u, x], pair[v, y]],
            t -> image[inverse[CATOFUNS], inverse[IMAGE[DUP]]]}] // Reverse

Out[24]= member[pair[pair[u, x], pair[v, y]], inv[CATOFUNS]] == and[equal[composite[u, v], id[x]],
           equal[composite[v, u], id[y]], member[u, map[y, x]], member[v, map[x, y]]]

In[25]:= member[pair[pair[u_, x_], pair[v_, y_]], inv[CATOFUNS]] :=
          and[equal[composite[u, v], id[x]], equal[composite[v, u], id[y]],
           member[u, map[y, x]], member[v, map[x, y]]]
```

The following diagram provides a picture for this membership rule.



This membership rule will not be needed after an explicit formula for **inv[CATOFUNS]** becomes available.

an explicit formula for inv[CATOFUNS]

Lemma.

```
In[26]:= SubstTest[implies, equal[r, s],
           equal[composite[r, t], composite[s, t]], {r → composite[funpart[u], funpart[v]],
            s → id[domain[funpart[v]]], t → inverse[funpart[v]]}] // Reverse

Out[26]= or[equal[composite[funpart[u], id[range[funpart[v]]]], inverse[funpart[v]]],
           not[equal[composite[funpart[u], funpart[v]], id[domain[funpart[v]]]]]] == True

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

Lemma.

```
In[28]:= Map[not, SubstTest[and, implies[p1, p3],
            implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 -> equal[composite[funpart[u], funpart[v]], id[domain[funpart[v]]]],
             p2 -> equal[domain[funpart[u]], range[funpart[v]]],
             p3 -> equal[composite[funpart[u], id[range[funpart[v]]]], inverse[funpart[v]]],
             p4 -> equal[funpart[u], inverse[funpart[v]]]}]] // Reverse

Out[28]= or[equal[funpart[u], inverse[funpart[v]]],
           not[equal[composite[funpart[u], funpart[v]], id[domain[funpart[v]]]]],
           not[equal[domain[funpart[u]], range[funpart[v]]]]] == True

In[29]:= (% /. {u → u_, v → v_}) /. Equal → SetDelayed
```

Theorem. (Removal of the **funpart** wrappers.)

```
In[30]:= SubstTest[implies, and[equal[x, funpart[u]], equal[y, funpart[v]]],
           or[equal[x, inverse[y]], not[equal[composite[x, y], id[domain[y]]]],
            not[equal[domain[x], range[y]]]], {u → x, v → y}] // Reverse

Out[30]= or[equal[x, inverse[y]], not[equal[composite[x, y], id[domain[y]]]],
           not[equal[domain[x], range[y]]], not[FUNCTION[x]], not[FUNCTION[y]]] == True

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

Lemma.

```
In[32]:= Map[not,
           SubstTest[and, implies[p1, p5], implies[and[p1, p5], p6], implies[and[p1, p6], p7],
            not[implies[p1, p7]], {p1 -> member[pair[pair[u, x], pair[v, y]], inv[CATOFUNS]],
             p5 → subclass[composite[v, u], Id], p6 -> equal[y, range[v]],
             p7 -> equal[domain[u], range[v]], p8 → equal[u, inverse[v]]}]] // Reverse

Out[32]= or[equal[domain[u], range[v]],
           not[equal[composite[u, v], id[x]]], not[equal[composite[v, u], id[y]]],
           not[member[u, map[y, x]]], not[member[v, map[x, y]]]] == True
```

```
In[33]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

The following steps are omitted in next derivation to expedite execution:

implies[p1, p2], **implies[p1, p3]**, **implies[p1, p4]**, **implies[p1, p7]**.

Theorem. If **pair[u, x]** and **pair[v, y]** are an inverse pair of morphisms in the category of sets, then **u** is the inverse of **v**.

```
In[34]:= Map[not, SubstTest[and, implies[and[p2, p3, p4, p7], p8], not[implies[p1, p8]],
           {p1 -> member[pair[pair[u, x], pair[v, y]], inv[CATOFUNS]], p2 → FUNCTION[u],
            p3 → FUNCTION[v], p4 → equal[composite[u, v], id[domain[v]]],
            p7 -> equal[domain[u], range[v]], p8 → equal[u, inverse[v]]}]] // Reverse

Out[34]= or[equal[u, inverse[v]],
           not[equal[composite[u, v], id[x]]], not[equal[composite[v, u], id[y]]],
           not[member[u, map[y, x]]], not[member[v, map[x, y]]]] == True

In[35]:= or[equal[u_, inverse[v_]],
           not[equal[composite[u_, v_], id[x_]]], not[equal[composite[v_, u_], id[y_]]],
           not[member[u_, map[y_, x_]]], not[member[v_, map[x_, y_]]]] := True
```

Theorem. (Remove variables.)

```
In[36]:= Map[composite[id[cart[V, V]], complement[#], id[cart[V, V]]] &,
          SubstTest[class, pair[pair[u, x], pair[v, y]],
           member[pair[pair[setpart[u], setpart[x]], pair[setpart[v], setpart[y]]], t],
           t -> complement[
             dif[inv[CATOFUNS], composite[inverse[FIRST], inverse[IMAGE[SWAP]], FIRST]]]]]

Out[36]= intersection[composite[inverse[FIRST], complement[inverse[IMAGE[SWAP]]], FIRST],
           inv[CATOFUNS]] == 0

In[37]:= % /. Equal → SetDelayed
```

Lemma. (Simplification rule.)

```
In[38]:= Assoc[inv[CATOFUNS], id[domain[inv[CATOFUNS]]], id[cart[V, V]]] // Reverse

Out[38]= composite[inv[CATOFUNS], id[cart[V, V]]] == inv[CATOFUNS]

In[39]:= % /. Equal → SetDelayed
```

Lemma. (Simplification rule.)

```
In[40]:= Assoc[id[cart[V, V]], id[range[inv[CATOFUNS]]], inv[CATOFUNS]]

Out[40]= composite[id[cart[V, V]], inv[CATOFUNS]] == inv[CATOFUNS]

In[41]:= % /. Equal → SetDelayed
```

Lemma.

```
In[42]:= SubstTest[empty, composite[id[t], dif[u, v], id[t]], {t → cart[V, V],
           u → inv[CATOFUNS], v -> composite[inverse[FIRST], inverse[IMAGE[SWAP]], FIRST]}]

Out[42]= subclass[inv[CATOFUNS], composite[inverse[FIRST], inverse[IMAGE[SWAP]], FIRST]] == True

In[43]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[44]:= SubstTest[subclass, inverse[s], intersection[t, inverse[t]], {s → inv[CATOFUNS],
           t -> composite[inverse[FIRST], inverse[IMAGE[SWAP]], FIRST]}] // Reverse

Out[44]= subclass[inv[CATOFUNS], composite[inverse[FIRST], INVERSE, FIRST]] == True

In[45]:= % /. Equal→ SetDelayed
```

Main Theorem. An explicit formula for **inv[CATOFUNS]**.

```
In[46]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]],
           {u -> inv[CATOFUNS], v -> composite[id[domain[inv[CATOFUNS]]],
              inverse[FIRST], INVERSE, FIRST, id[domain[inv[CATOFUNS]]]]}] // Reverse

Out[46]= equal[composite[id[composite[IMAGE[SECOND], id[BIJ]]], inverse[FIRST], INVERSE,
           FIRST, id[composite[IMAGE[SECOND], id[BIJ]]]], inv[CATOFUNS]] == True

In[47]:= inv[CATOFUNS] := composite[id[composite[IMAGE[SECOND], id[BIJ]]],
           inverse[FIRST], INVERSE, FIRST, id[composite[IMAGE[SECOND], id[BIJ]]]]
```

Comment. The temporary membership rule for **inv[CATOFUNS]** is now automatically replaced by a simpler one:

```
In[48]:= member[pair[pair[u, x], pair[v, y]], inv[CATOFUNS]]

Out[48]= and[equal[v, inverse[u]], member[u, bij[domain[u], x]],
          member[v, bij[domain[v], y]], member[x, V], member[y, V]]
```