neutral elements for CATORELN

Johan G. F. Belinfante
2006 September 17

```
In[1]:= SetDirectory["l:"]; << goedel85.17a; << tools.m

        :Package Title: goedel85.17a          2006 September 17 at 10:25 a.m.

        It is now:  2006 Sep 17 at 12:54

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 August 22

        weightlimit = 40
```

summary

It is shown that identity morphisms and non-morphisms for the category of relations are left and right neutral elements, and that there are no others.

non-morphisms are left and right neutral

For convenience, a temporary abbreviation is introduced for the class of morphisms of the category of relations:

```
In[2]:= MOR := composite[S, IMAGE[SECOND], id[P[cart[V, V]]]]
```

The condition for **x** to be a morphism is:

```
In[3]:= member[x, MOR]

Out[3]= subclass[first[x], cart[V, second[x]]]
```

Non-morphisms are right-neutral:

```
In[4]:= SubstTest[implies, equal[0, z], subclass[z, Id], z → composite[CATORELN, RIGHT[x]]]

Out[4]= or[subclass[composite[CATORELN, RIGHT[x]], Id],
          subclass[first[x], cart[V, second[x]]]] == True

In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= SubstTest[implies, member[domain[w], V], not[equal[w, V]], w → first[x]]

Out[6]= or[member[first[x], V], not[member[domain[first[x]], V]]] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

Simplification rule.

```
In[8]:= equiv[member[domain[first[x]], V], member[first[x], V]]

Out[8]= True

In[9]:= member[domain[first[x_]], V] := member[first[x], V]
```

Non-morphisms are left-neutral:

```
In[10]:= SubstTest[implies, equal[0, z], subclass[z, Id], z -> composite[CATORELN, LEFT[x]]]

Out[10]= or[subclass[composite[CATORELN, LEFT[x]], Id],
           subclass[first[x], cart[V, second[x]]]] == True

In[11]:= (% /. x → x_) /. Equal → SetDelayed
```

---

identity and inclusion morphisms

An **identity morphism** is an element of the form **PAIR[id[x], x]**. A temporary abbreviation is introduced for the class of identity morphisms:

```
In[12]:= IDS := inverse[IMAGE[DUP]]
```

The condition for x is be an identity morphism is:

```
In[13]:= member[x, IDS]

Out[13]= equal[first[x], id[second[x]]]
```

An **inclusion morphism** is an element of the form **PAIR[id[x], y]**, where **x** is a subset of **y**. The class of inclusion morphisms is

```
In[14]:= class[pair[x, y], exists[t, and[equal[x, id[t]], subclass[t, y]]]]

Out[14]= composite[inverse[IMAGE[DUP]], S]
```

The condition for **x** to be an inclusion morphism is:

```
In[15]:= member[x, composite[inverse[IMAGE[DUP]], S]]

Out[15]= subclass[first[x], id[second[x]]]
```

Theorem. An inclusion is an identity if **domain[first[x]] = second[x]**.

```
In[16]:= equiv[and[equal[y, domain[x]], subclass[x, id[y]]], equal[x, id[y]]]

Out[16]= True

In[17]:= and[equal[y_, domain[x_]], subclass[x_, id[y_]]] := equal[x, id[y]]
```

left-neutral inclusions are identities

Lemma.

```
In[18]:= member[x, fix[composite[y, inverse[S], IMAGE[FIRST]]]] // AssertTest

Out[18]= member[x, fix[composite[y, inverse[S], IMAGE[FIRST]]]] ==
          member[x, image[y, P[domain[x]]]]

In[19]:= member[x_, fix[composite[y_, inverse[S], IMAGE[FIRST]]]] :=
          member[x, image[y, P[domain[x]]]]
```

Lemma.

```
In[20]:= member[id[x], image[
            inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], P[cart[V, x]]]] // AssertTest

Out[20]= member[id[x],
           image[inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], P[cart[V, x]]]] == False

In[21]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Inclusion morphisms are left-neutral if and only if they are identities.

```
In[22]:= Map[not[member[PAIR[id[x], y], domain[#]]] &, SubstTest[fix,
           composite[inverse[SECOND], Di, direct[COMPOSE, w], id[domain[CATORELN]]], w → FIRST]]

Out[22]= subclass[composite[CATORELN, LEFT[PAIR[id[x], y]]], Id] ==
          or[equal[x, y], not[member[y, V]], not[subclass[x, y]]]

In[23]:= subclass[composite[CATORELN, LEFT[PAIR[id[x_], y_]]], Id] :=
          or[equal[x, y], not[member[y, V]], not[subclass[x, y]]]
```

In particular, identity morphisms are left-neutral:

```
In[24]:= subclass[composite[CATORELN, LEFT[PAIR[id[x], x]]], Id]

Out[24]= True
```

A **PAIR**-free restatement of this is:

```
In[25]:= SubstTest[implies, and[equal[u, id[v]], equal[x, PAIR[u, v]]],
           subclass[composite[CATORELN, LEFT[x]], Id], {u → first[x], v → second[x]}] // MapNotNot

Out[25]= or[not[equal[first[x], id[second[x]]]],
           subclass[composite[CATORELN, LEFT[x]], Id]] == True
```

```
In[26]:= (% /. x → x_) /. Equal → SetDelayed
```

The statement that left-neutral inclusions are identities can also be restated without **PAIR**.

```
In[27]:= SubstTest[implies, and[equal[s, id[u]],
           equal[x, PAIR[s, t]], subclass[composite[CATORELN, LEFT[x]], Id]],
         or[equal[u, t], not[member[t, V]], not[subclass[u, t]]],
         {s → first[x], t → second[x], u → domain[first[x]]}] // MapNotNot

Out[27]= or[equal[domain[first[x]], second[x]],
           not[subclass[composite[CATORELN, LEFT[x]], Id]],
           not[subclass[first[x], id[second[x]]]]] == True

In[28]:= or[equal[domain[first[x_]], second[x_]],
           not[subclass[composite[CATORELN, LEFT[x_]], Id]],
           not[subclass[first[x_], id[second[x_]]]]] := True
```

Theorem. Left-neutral inclusions are identity morphisms.

```
In[29]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p1, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 -> member[x, composite[inverse[IMAGE[DUP]], S]],
            p2 -> subclass[composite[CATORELN, LEFT[x]], Id],
            p3 -> equal[domain[first[x]], second[x]], p4 → member[x, IDS]}]]

Out[29]= or[equal[first[x], id[second[x]]], not[subclass[composite[CATORELN, LEFT[x]], Id]],
           not[subclass[first[x], id[second[x]]]]] == True

In[30]:= or[equal[first[x_], id[second[x_]]], not[subclass[composite[CATORELN, LEFT[x_]], Id]],
           not[subclass[first[x_], id[second[x_]]]]] := True
```

---

identities are right-neutral

Lemma.

```
In[31]:= Map[member[x, #] &, fix[composite[IMAGE[FIRST], id[P[cart[V, V]]],
             inverse[fix[composite[inverse[FIRST], Di, COMPOSE]]], IMAGE[DUP]]] // Renormality]

Out[31]= member[pair[id[x], x], composite[IMAGE[FIRST], id[P[cart[V, V]]],
           inverse[fix[composite[inverse[FIRST], Di, COMPOSE]]]]] == False

In[32]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Identity morphisms are right-neutral.

```
In[33]:= Map[not[member[w, range[#]]] &, SubstTest[fix, composite[inverse[FIRST], Di,
             direct[COMPOSE, w], id[domain[CATORELN]]], w → FIRST]] /. w -> PAIR[id[x], x]

Out[33]= subclass[composite[CATORELN, RIGHT[PAIR[id[x], x]]], Id] == True

In[34]:= (% /. x → x_) /. Equal → SetDelayed
```

The same result can be restated without **PAIR** as follows:

```
In[35]:= SubstTest[implies, and[equal[x, PAIR[s, t]], equal[s, id[t]]],
          subclass[composite[CATORELN, RIGHT[x]], Id], {s → first[x], t → second[x]}]

Out[35]= or[not[equal[first[x], id[second[x]]]],
           subclass[composite[CATORELN, RIGHT[x]], Id]] == True

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

there are no other right neutral elements

The following general uniqueness theorem for left and right neutral elements will be used to show that there are no other right-neutral elements other than the non-morphisms and identity morphisms.

```
In[37]:= implies[and[member[u, V], member[v, V], subclass[composite[x, LEFT[u]], Id],
           subclass[composite[x, RIGHT[v]], Id], member[pair[u, v], domain[x]]], equal[u, v]]

Out[37]= True
```

Lemma.

```
In[38]:= (SubstTest[implies, and[member[u, V], member[v, V], subclass[composite[x, LEFT[u]], Id],
              subclass[composite[x, RIGHT[v]], Id], member[pair[u, v], domain[x]]],
            equal[u, v], x → CATORELN] /. u → PAIR[id[second[v]], second[v]]) // MapNotNot

Out[38]= or[equal[v, PAIR[id[second[v]], second[v]]],
           not[subclass[composite[CATORELN, RIGHT[v]], Id]],
           not[subclass[first[v], cart[V, second[v]]]]] == True

In[39]:= (% /. v → v_) /. Equal → SetDelayed
```

Theorem. Every right-neutral morphism is an identity morphism.

```
In[40]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → member[x, MOR], p2 → subclass[composite[CATORELN, RIGHT[x]], Id],
            p3 -> equal[x, PAIR[id[second[x]], second[x]]], p4 → member[x, IDS]}]]

Out[40]= or[equal[first[x], id[second[x]]], not[subclass[composite[CATORELN, RIGHT[x]], Id]],
           not[subclass[first[x], cart[V, second[x]]]]] == True

In[41]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining all these results, one finds:

```
In[42]:= equiv[subclass[composite[CATORELN, RIGHT[x]], Id], or[equal[first[x], id[second[x]]],
            not[subclass[first[x], cart[V, second[x]]]]]] // not // not

Out[42]= True
```

```
In[43]:= subclass[composite[CATORELN, RIGHT[x_]], Id] :=
          or[equal[first[x], id[second[x]]], not[subclass[first[x], cart[V, second[x]]]]]
```

Variable-free reformulation:

```
In[44]:= range[fix[composite[inverse[FIRST], Di, CATORELN]]] // Normality

Out[44]= range[fix[composite[inverse[FIRST], Di, CATORELN]]] ==
          composite[intersection[complement[inverse[IMAGE[DUP]]], composite[S, IMAGE[SECOND]]],
           id[P[cart[V, V]]]]

In[45]:= range[fix[composite[inverse[FIRST], Di, CATORELN]]] :=
          composite[intersection[complement[inverse[IMAGE[DUP]]],
            composite[S, IMAGE[SECOND]]], id[P[cart[V, V]]]]
```

left-neutral elements of a restriction of COMPOSE

The class of left-neutral elements of **composite[COMPOSE, id[cart[V, x]]]** is

```
In[46]:= class[y, subclass[composite[w, LEFT[y]], Id]] /. w → composite[COMPOSE, id[cart[V, x]]]

Out[46]= complement[image[inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], x]]
```

In this section a membership rule for this class is derived using some tricky maneuvering. The first lemma uses **AssertTest**, with **setpart** being used to clean up the result that is obtained.

```
In[47]:= (member[t, image[inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], y]] //
            AssertTest) /. t → setpart[x]

Out[47]= member[setpart[x], image[inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], y]] ==
          not[subclass[y, fix[IMAGE[cross[Id, setpart[x]]]]]]

In[48]:= member[setpart[x_],
           image[inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], y_]] :=
          not[subclass[y, fix[IMAGE[cross[Id, setpart[x]]]]]]
```

Next the **setpart** wrapper is eliminated.

```
In[49]:= (SubstTest[implies, equal[x, setpart[t]],
             equiv[member[x, image[inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], y]],
              not[subclass[y, fix[IMAGE[cross[Id, x]]]]]],
             t → x] /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[50]:= equiv[member[x, image[inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], y]],
           and[member[x, V], not[subclass[y, fix[IMAGE[cross[Id, x]]]]]]]

Out[50]= True
```

```
In[51]:= member[x_, image[inverse[fix[composite[inverse[SECOND], Di, COMPOSE]]], y_]] :=
          and[member[x, V], not[subclass[y, fix[IMAGE[cross[Id, x]]]]]]
```

there are no other left-neutral elements

In this section it is shown that left-neutral morphisms in the category of relations are inclusion morphisms. The basic idea is to use that fact that **CATORELN** is a restriction of the direct product of **COMPOSE** and **FIRST**.

```
In[52]:= SubstTest[implies, and[member[r, s], subclass[s, t]], member[r, t],
          {r -> set[PAIR[u, v]], s -> P[cart[V, domain[x]]], t -> fix[IMAGE[cross[Id, x]]]}]

Out[52]= or[and[member[v, V], not[member[v, domain[x]]]],
           equal[image[x, set[v]], set[v]], not[member[u, V]],
           not[subclass[P[cart[V, domain[x]]], fix[IMAGE[cross[Id, x]]]]]] == True

In[53]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

The variables **u** and **v** are removed, and **setpart** is used to simplify the final result.

```
In[54]:= (Map[equal[0, composite[#, complement[#]]] &, SubstTest[class, pair[u, v],
              or[and[member[v, V], not[member[v, domain[y]]]], equal[image[y, set[v]], set[v]],
               not[member[u, V]], not[subclass[s, t]], {s -> P[cart[V, domain[y]]],
               t -> fix[IMAGE[cross[Id, y]]]}]] // Reverse) /. y → setpart[x]

Out[54]= or[not[subclass[P[cart[V, domain[setpart[x]]]], fix[IMAGE[cross[Id, setpart[x]]]]]],
           subclass[composite[Id, setpart[x]], Id]] == True

In[55]:= (% /. x → x_) /. Equal → SetDelayed
```

Removing the **setpart** wrapper yields:

```
In[56]:= SubstTest[implies, equal[x, setpart[y]],
          or[not[subclass[P[cart[V, domain[x]]], fix[IMAGE[cross[Id, x]]]]],
           subclass[composite[Id, x], Id]], y → x]

Out[56]= or[not[member[x, V]], not[subclass[P[cart[V, domain[x]]], fix[IMAGE[cross[Id, x]]]]],
           subclass[composite[Id, x], Id]] == True

In[57]:= (% /. x → x_) /. Equal → SetDelayed
```

The fact that **CATORELN** is a restriction of **direct[COMPOSE, FIRST]** inplies:

```
In[58]:= Map[implies[and[member[u, V], member[v, V], #],
              or[not[subclass[u, cart[V, v]]], subclass[composite[Id, u], Id]]] &,
             Map[not[member[PAIR[u, v], domain[#]]] &, SubstTest[fix, composite[inverse[SECOND],
               Di, direct[COMPOSE, w], id[domain[CATORELN]]], w → FIRST]] // MapNotNot

Out[58]= or[not[member[u, V]], not[member[v, V]], not[subclass[u, cart[V, v]]],
           not[subclass[composite[CATORELN, LEFT[PAIR[u, v]]], Id]],
           subclass[composite[Id, u], Id]] == True
```

```
In[59]:= (% /. {u → u_, v → v_}) /. Equal → SetDelayed
```

The **PAIR[u,v]** can be replaced with a single variable **x** as follows:

```
In[60]:= SubstTest[implies, equal[x, PAIR[u, v]],
           or[not[member[u, V]], not[member[v, V]], not[subclass[u, cart[V, v]]],
            not[subclass[composite[CATORELN, LEFT[x]], Id]], subclass[composite[Id, u], Id]],
           {u → first[x], v → second[x]}] // MapNotNot

Out[60]= or[not[subclass[composite[CATORELN, LEFT[x]], Id]],
           not[subclass[first[x], cart[V, second[x]]]],
           subclass[composite[Id, first[x]], Id]] == True

In[61]:= (% /. x → x_) /. Equal → SetDelayed
```

This result can be cleaned up as follows:

```
In[62]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           not[implies[and[p1, p2], p4]], {p1 → subclass[composite[CATORELN, LEFT[x]], Id],
            p2 -> subclass[first[x], cart[V, second[x]]],
            p3 -> subclass[composite[Id, first[x]], Id], p4 → subclass[first[x], Id]}]]

Out[62]= or[not[subclass[composite[CATORELN, LEFT[x]], Id]],
           not[subclass[first[x], cart[V, second[x]]]], subclass[first[x], Id]] == True

In[63]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[87]:= or[not[equal[domain[first[x]], range[first[x]]]],
           not[subclass[first[x], Id]], not[subclass[range[first[x]], second[x]]],
           subclass[first[x], id[second[x]]]] // NotNotTest

Out[87]= or[not[equal[domain[first[x]], range[first[x]]]], not[subclass[first[x], Id]],
           not[subclass[range[first[x]], second[x]]], subclass[first[x], id[second[x]]]] == True

In[88]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[89]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p2, p4], implies[p3, p5],
           implies[and[p3, p4, p5], p6], implies[and[p1, p6], p7], implies[and[p3, p7], p8],
           not[implies[and[p1, p2], p8]], {p1 -> subclass[composite[CATORELN, LEFT[x]], Id],
            p2 -> subclass[first[x], cart[V, second[x]]], p3 -> subclass[first[x], Id],
            p4 → subclass[range[first[x]], second[x]], p5 →
             equal[domain[first[x]], range[first[x]]], p6 -> subclass[first[x], id[second[x]]],
            p7 → equal[domain[first[x]], second[x]], p8 -> equal[first[x], id[second[x]]]}]]

Out[89]= or[equal[first[x], id[second[x]]], not[subclass[composite[CATORELN, LEFT[x]], Id]],
           not[subclass[first[x], cart[V, second[x]]]]] == True

In[90]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[92]:= equiv[subclass[composite[CATORELN, LEFT[x]], Id],
            or[not[subclass[first[x], cart[V, second[x]]]],
             equal[first[x], id[second[x]]]]] // not // not

Out[92]= True

In[93]:= subclass[composite[CATORELN, LEFT[x_]], Id] :=
          or[equal[first[x], id[second[x]]], not[subclass[first[x], cart[V, second[x]]]]]
```

Variable-free restatement:

```
In[94]:= domain[fix[composite[inverse[SECOND], Di, CATORELN]]] // Normality

Out[94]= domain[fix[composite[inverse[SECOND], Di, CATORELN]]] ==
          composite[intersection[complement[inverse[IMAGE[DUP]]], composite[S, IMAGE[SECOND]]],
           id[P[cart[V, V]]]]

In[95]:= domain[fix[composite[inverse[SECOND], Di, CATORELN]]] :=
          composite[intersection[complement[inverse[IMAGE[DUP]]],
            composite[S, IMAGE[SECOND]]], id[P[cart[V, V]]]]
```