FUNCTION[id[V × V] ∘ inverse[x]]

Johan G. F. Belinfante
2011 July 15

```
In[1]:= SetDirectory["l:"]; << goedel.11jul13a

:Package Title: goedel.11jul13a                 2011 July 13 at 3:20 p.m.

Loading takes about eleven minutes, half that time due to builtin pauses.

It is now:  2011 Jul 15 at 7:39

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2011 Jul 15 at 7:50
```

summary

In this notebook it is shown that if **FIRST ∘ inverse[x]** and **rotate[x]** are both functions, then so is **id[V × V] ∘ inverse[x]**, and conversely.

fu-1-2.nb

A key ingredient in the derivation is the following rewrite rule, which was derived in the posted notebook **fu-1-2.nb** dated 2006 August 27.

```
In[2]:= and[FUNCTION[composite[SECOND, x]], FUNCTION[composite[FIRST, x]]]

Out[2]= FUNCTION[composite[id[cart[V, V]], x]]
```

The following immediate corollary of this result will be needed in the sequel.

Corollary.

```
In[3]:= or[FUNCTION[composite[id[cart[V, V]], x]], not[FUNCTION[composite[FIRST, x]]],
          not[FUNCTION[composite[SECOND, x]]]] // NotNotTest

Out[3]= or[FUNCTION[composite[id[cart[V, V]], x]],
          not[FUNCTION[composite[FIRST, x]]], not[FUNCTION[composite[SECOND, x]]]] == True
```

```
In[4]:= or[FUNCTION[composite[id[cart[V, V]], x_]],
          not[FUNCTION[composite[FIRST, x_]]], not[FUNCTION[composite[SECOND, x_]]]] := True
```

derivation

In the derivation to be presented, five additional variables are introduced for two horizontally separated points belonging to the graph of a ternary relation **x**, and later these variables are eliminated. Each of the points is an ordered triplet, sharing the same third coordinate.

Lemma. A consequence of the condition that **FIRST** ∘ **inverse[x]** be a function.

```
In[9]:= (SubstTest[implies, and[member[pair[s, t], composite[Id, y]],
            member[pair[r, s], composite[Id, z]]], member[pair[r, t], composite[y, z]],
           {y → FIRST, z → inverse[x], r → w, s → pair[u, v]}] // Reverse) /. t → u

Out[9]= or[member[pair[w, u], composite[FIRST, inverse[x]]], not[member[u, V]],
          not[member[v, V]], not[member[w, V]], not[member[pair[pair[u, v], w], x]]] == True

In[10]:= (% /. {u → u_, v → v_, w → w_, x → x_}) /. Equal → SetDelayed
```

Lemma. Equality of the first coordinates of the two points.

```
In[12]:= Map[not, SubstTest[and, implies[and[p0, p1], p5], implies[and[p0, p2], p6],
            implies[and[p0, p3, p5, p6], p7], not[implies[and[p0, p1, p2, p3], p7]],
            {p0 → and[member[s, V], member[t, V], member[u, V], member[v, V], member[w, V]],
             p1 -> member[pair[pair[s, t], w], x], p2 -> member[pair[pair[u, v], w], x],
             p3 -> FUNCTION[composite[FIRST, inverse[x]]],
             p5 -> member[pair[w, s], composite[FIRST, inverse[x]]],
             p6 -> member[pair[w, u], composite[FIRST, inverse[x]]], p7 → equal[s, u]}]] // Reverse

Out[12]= or[equal[s, u], not[FUNCTION[composite[FIRST, inverse[x]]]], not[member[s, V]],
           not[member[t, V]], not[member[u, V]], not[member[v, V]], not[member[w, V]],
           not[member[pair[pair[s, t], w], x]], not[member[pair[pair[u, v], w], x]]] == True

In[13]:= (% /. {s → s_, t → t_, u → u_, v → v_, w → w_, x → x_}) /. Equal → SetDelayed
```

Lemma. The condition that **rotate[x]** be a function implies equality of the second coordinates when the first and third coordinates are equal.

```
In[16]:= SubstTest[implies, and[FUNCTION[s], member[pair[r, t], s], member[pair[r, v], s]],
           equal[t, v], {s → rotate[x], r → pair[w, u]}] // Reverse

Out[16]= or[equal[t, v], not[FUNCTION[rotate[x]]], not[member[t, V]],
           not[member[u, V]], not[member[v, V]], not[member[w, V]],
           not[member[pair[pair[u, t], w], x]], not[member[pair[pair[u, v], w], x]]] == True

In[17]:= (% /. {t → t_, u → u_, v → v_, w → w_, x → x_}) /. Equal → SetDelayed
```

Lemma. When both conditions on **x** hold, the two points have the same second coordinates.

```
In[19]:= Map[not, SubstTest[and, implies[and[p0, p1, p2, p3], p5],
           implies[and[p0, p1, p2, p4, p5], p6], not[implies[and[p0, p1, p2, p3, p4], p6]],
           {p0 → and[member[s, V], member[t, V], member[u, V], member[v, V], member[w, V]],
            p1 -> member[pair[pair[s, t], w], x], p2 -> member[pair[pair[u, v], w], x],
            p3 -> FUNCTION[composite[FIRST, inverse[x]]], p4 → FUNCTION[rotate[x]],
            p5 → equal[s, u], p6 → equal[t, v]}]] // Reverse

Out[19]= or[equal[t, v], not[FUNCTION[composite[FIRST, inverse[x]]]],
          not[FUNCTION[rotate[x]]], not[member[s, V]], not[member[t, V]],
          not[member[u, V]], not[member[v, V]], not[member[w, V]],
          not[member[pair[pair[s, t], w], x]], not[member[pair[pair[u, v], w], x]]] == True

In[20]:= (% /. {s → s_, t → t_, u → u_, v → v_, w → w_, x → x_}) /. Equal → SetDelayed
```

Lemma. A simplification rule.

```
In[22]:= SubstTest[subclass, composite[t, inverse[t]],
          Id, t → composite[x, inverse[y]]] // Reverse

Out[22]= subclass[composite[x, inverse[y], y, inverse[x]], Id] ==
         FUNCTION[composite[x, inverse[y]]]

In[23]:= subclass[composite[x_, inverse[y_], y_, inverse[x_]], Id] :=
         FUNCTION[composite[x, inverse[y]]]
```

Theorem. Eliminating the five variables for the two points.

```
In[25]:= Map[empty[domain[#]] &,
         Map[composite[complement[#], id[cart[cart[V, V], cart[V, V]]]] &,
          SubstTest[class, pair[pair[pair[s, t], pair[u, v]], w],
           or[equal[t, v], not[FUNCTION[y]], not[FUNCTION[z]], not[member[s, V]],
            not[member[t, V]], not[member[u, V]], not[member[v, V]], not[member[w, V]],
            not[member[pair[pair[s, t], w], x]], not[member[pair[pair[u, v], w], x]]],
           {y → rotate[x], z -> composite[FIRST, inverse[x]]}]]]

Out[25]= or[FUNCTION[composite[SECOND, inverse[x]]],
          not[FUNCTION[composite[FIRST, inverse[x]]]], not[FUNCTION[rotate[x]]]] == True

In[27]:= or[FUNCTION[composite[SECOND, inverse[x_]]],
          not[FUNCTION[composite[FIRST, inverse[x_]]]], not[FUNCTION[rotate[x_]]]] := True
```

At this point the corollary derived in the first section is used.

Main Theorem. If **FIRST ∘ inverse[x]** and **rotate[x]** are both functions, then **id[V × V] ∘ inverse[x]** is a function.

```
In[29]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 -> and[FUNCTION[composite[FIRST, inverse[x]]], FUNCTION[rotate[x]]],
             p2 -> FUNCTION[composite[FIRST, inverse[x]]],
             p3 -> FUNCTION[composite[SECOND, inverse[x]]],
             p4 -> FUNCTION[composite[id[cart[V, V]], inverse[x]]]}]] // Reverse

Out[29]= or[FUNCTION[composite[id[cart[V, V]], inverse[x]]],
           not[FUNCTION[composite[FIRST, inverse[x]]]], not[FUNCTION[rotate[x]]]] == True

In[30]:= or[FUNCTION[composite[id[cart[V, V]], inverse[x_]]],
           not[FUNCTION[composite[FIRST, inverse[x_]]]], not[FUNCTION[rotate[x_]]]] := True
```

converses and corollaries

The converse of the main theorem is a trivial consequence of the following two lemmas.

Lemma. (An implication in the reverse direction.)

```
In[31]:= SubstTest[implies, FUNCTION[t], FUNCTION[rotate[inverse[t]]],
           t -> composite[id[cart[V, V]], inverse[x]]] // Reverse

Out[31]= or[FUNCTION[rotate[x]], not[FUNCTION[composite[id[cart[V, V]], inverse[x]]]]] == True

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

Lemma. (Another implication in the reverse direction.)

```
In[34]:= SubstTest[implies, and[p, q], p,
           {p -> FUNCTION[composite[FIRST, x]], q -> FUNCTION[composite[SECOND, x]]}] // Reverse

Out[34]= or[FUNCTION[composite[FIRST, x]], not[FUNCTION[composite[id[cart[V, V]], x]]]] == True

In[35]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Combining the main theorem with its converse.

```
In[36]:= equiv[FUNCTION[composite[id[cart[V, V]], inverse[x]]],
            and[FUNCTION[composite[FIRST, inverse[x]]], FUNCTION[rotate[x]]]] // not // not

Out[36]= True

In[37]:= and[FUNCTION[composite[FIRST, inverse[x_]]], FUNCTION[rotate[x_]]] :=
          FUNCTION[composite[id[cart[V, V]], inverse[x]]]
```

Corollary. A rewrite rule similar to the preceding one, but with **x** replaced with its inverse.

```
In[38]:= SubstTest[and, FUNCTION[composite[FIRST, inverse[t]]],
           FUNCTION[rotate[t]], t → inverse[x]] // Reverse

Out[38]= and[FUNCTION[composite[FIRST, x]], FUNCTION[rotate[inverse[x]]]] ==
          FUNCTION[composite[id[cart[V, V]], x]]

In[39]:= and[FUNCTION[composite[FIRST, x_]], FUNCTION[rotate[inverse[x_]]]] :=
          FUNCTION[composite[id[cart[V, V]], x]]
```