binary isomorphisms

Johan G. F. Belinfante
2006 December 29

```
In[1]:= SetDirectory["l:"]; << goedel88.26a; << tools.m

        :Package Title: goedel88.26a        2006 December 26 at 3:50 p.m.

        It is now:  2006 Dec 29 at 12:54

        Loading Simplification Rules

        TOOLS.M                        Revised 2006 December 17

        weightlimit = 40
```

summary

If a binary homomorphism of binary operations is one-to-one and onto, then its inverse is also a binary homomorphism.

derivation

The main idea is pretty simple, but some cleaning up will be required.

```
In[2]:= SubstTest[implies, equal[u, v],
          equal[composite[t, u, cross[t, t]], composite[t, v, cross[t, t]]],
          {u -> composite[oopart[w], x], v -> composite[y, cross[oopart[w], oopart[w]]],
           t → inverse[oopart[w]]}] // Reverse

Out[2]= or[equal[
           composite[id[domain[oopart[w]]], x, cross[inverse[oopart[w]], inverse[oopart[w]]]],
           composite[inverse[oopart[w]], y, id[cart[range[oopart[w]], range[oopart[w]]]]]],
          not[equal[composite[y, cross[oopart[w], oopart[w]]], composite[oopart[w], x]]]] == True

In[3]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

The **oopart** wrapper can be removed.

```
In[4]:= SubstTest[implies,
          and[equal[w, oopart[t]], equal[composite[y, cross[w, w]], composite[w, x]]],
          equal[composite[id[domain[w]], x, cross[inverse[w], inverse[w]]],
           composite[inverse[w], y, id[cart[range[w], range[w]]]]], t → w] // Reverse

Out[4]= or[equal[composite[id[domain[w]], x, cross[inverse[w], inverse[w]]],
           composite[inverse[w], y, id[cart[range[w], range[w]]]]],
          not[equal[composite[w, x], composite[y, cross[w, w]]]],
          not[FUNCTION[w]], not[FUNCTION[inverse[w]]]] == True

In[5]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4], implies[and[p3, p4], p5],
           not[implies[and[p1, p2], p5]], {p1 -> member[w, binhom[x, y]], p2 -> member[x, BINOPS],
            p3 → equal[domain[w], fix[domain[x]]], p4 → subclass[range[x], fix[domain[x]]],
            p5 → subclass[range[x], domain[w]]}]] // Reverse

Out[6]= or[not[member[w, binhom[x, y]]],
          not[member[x, BINOPS]], subclass[range[x], domain[w]]] == True

In[7]:= or[not[member[w_, binhom[x_, y_]]],
          not[member[x_, BINOPS]], subclass[range[x_], domain[w_]]] := True
```

The factor **id[domain[w]]** can now be removed:

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p1, p6], implies[and[p2, p3, p4], p5], implies[and[p5, p6], p7],
           not[implies[and[p1, p4], p7]], {p1 → and[member[w, binhom[x, y]], member[x, BINOPS]],
            p2 → FUNCTION[w], p3 -> equal[composite[w, x], composite[y, cross[w, w]]],
            p4 → FUNCTION[inverse[w]], p5 -> equal[composite[id[domain[w]],
               x, cross[inverse[w], inverse[w]]], composite[inverse[w], y,
               id[cart[range[w], range[w]]]]], p6 → subclass[range[x], domain[w]],
            p7 -> equal[composite[x, cross[inverse[w], inverse[w]]],
              composite[inverse[w], y, id[cart[range[w], range[w]]]]]}]] // Reverse

Out[8]= or[equal[composite[x, cross[inverse[w], inverse[w]]],
           composite[inverse[w], y, id[cart[range[w], range[w]]]]], not[FUNCTION[inverse[w]]],
          not[member[w, binhom[x, y]]], not[member[x, BINOPS]]] == True

In[9]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Main theorem. If a binary homomorphism of binary operations is one-to-one and onto, then its inverse is also a binary homomorphism. Comment. The execution time of this derivation was cut in half by omitting the following steps of the complete proof: **implies[p1, p4]**, **implies[and[p1, p4], p5]**, **implies[p1, p6]**, **implies[and[p3, p7], p8]**. The rewrite rules of the GOEDEL program automatically supply these omitted steps.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[and[p1, p2], p3], implies[and[p1, p5, p6], p7], not[implies[p1, p8]],
            {p1 → and[member[x, BINOPS], member[y, BINOPS], member[w, binhom[x, y]],
               FUNCTION[inverse[w]], equal[range[w], fix[domain[y]]]],
             p2 → member[w, map[fix[domain[x]], fix[domain[y]]]],
             p3 → member[inverse[w], map[fix[domain[y]], fix[domain[x]]]],
             p4 → equal[composite[w, x], composite[y, cross[w, w]]],
             p5 → equal[composite[x, cross[inverse[w], inverse[w]]],
               composite[inverse[w], y, id[cart[range[w], range[w]]]]],
             p6 → equal[domain[y], cartsq[fix[domain[y]]]],
             p7 ->
              equal[composite[x, cross[inverse[w], inverse[w]]], composite[inverse[w], y]],
             p8 → member[inverse[w], binhom[y, x]]}]] // Reverse

Out[10]= or[member[inverse[w], binhom[y, x]], not[equal[fix[domain[y]], range[w]]],
           not[FUNCTION[inverse[w]]], not[member[w, binhom[x, y]]],
           not[member[x, BINOPS]], not[member[y, BINOPS]]] == True

In[12]:= or[member[inverse[w_], binhom[y_, x_]], not[equal[fix[domain[y_]], range[w_]]],
           not[FUNCTION[inverse[w_]]], not[member[w_, binhom[x_, y_]]],
           not[member[x_, BINOPS]], not[member[y_, BINOPS]]] := True
```