binary homomorphisms of groups preserve inverses

Johan G. F. Belinfante
2011 December 19

```
In[1]:= SetDirectory["l:"]; << goedel.11dec18a

        :Package Title: goedel.11dec18a              2011 December 18 at 6:30 p.m.

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

        It is now:  2011 Dec 19 at 21:53

        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 Dec 19 at 22:6
```

summary

It is shown in this notebook that binary homomorphisms between groups preserve inverses. The entire derivation avoids working with individual group elements, and takes advantage of the fact that for groups, one-sided inverses are in fact two-sided. From a technical standpoint, what is shown first is that binary homomorphisms are monotone with respect to inversion. Two separate monotonicity inclusions are derived. In addition to these inclusions, an equation is also derived that says that binary homomorphisms between groups intertwine with the inversion functions for the group. That is, for any element of the first group, the inverse of its transform under the homomorphism is the transform of its inverse.

derivation

The concept of binary homomorphism comes from the general theory of binary operations. Binary homomorphisms are mappings from the fixed point class of the domain of one binary operation to that of another. Since the domain of a group is the cartesian square of its range, the fixed point set of its domain is the same as its range, and it is more natural to talk about the latter.

Lemma. The domain of a binary homomorphism from a group **x** to any binary operation **y** is equal to the range of **x**.

```
In[2]:= Map[not, SubstTest[and, implies[p2, p3], not[implies[and[p1, p2], p4]],
          {p1 → member[x, GROUPS], p2 → member[t, binhom[x, y]],
           p3 → equal[domain[t], fix[domain[x]]], p4 → equal[domain[t], range[x]]}]] // Reverse

Out[2]= or[equal[domain[t], range[x]],
          not[member[t, binhom[x, y]]], not[member[x, GROUPS]]] == True

In[3]:= or[equal[domain[t_], range[x_]],
          not[member[t_, binhom[x_, y_]]], not[member[x_, GROUPS]]] := True
```

A binary homomorphism **t** from a group **x** to a group **y** preserves the neutral element. Instead of saying that applying the function **t** to the neutral element **e[x] ∈ range[x]** yields the neutral element **e[y] ∈ range[y]**, one can equivalently say that the ordered pair of the two neutral elements is a point on the graph of **t**.

Theorem. If **t** is a binary homomorphism from a group **x** to a group **y**, then **pair[e[x], e[y]] ∈ t**.

```
In[4]:= Map[not, SubstTest[and, implies[p3, p4], implies[and[p1, p3], p5],
           implies[and[p1, p2, p3], p6], implies[and[p1, p5], p7],
           (* implies[and[p4,p6,p7],p8], *) not[implies[and[p1, p2, p3], p8]],
           {p1 → member[x, GROUPS], p2 → member[y, GROUPS], p3 → member[t, binhom[x, y]],
            p4 → FUNCTION[t], p5 → equal[domain[t], range[x]], p6 → equal[APPLY[t, e[x]], e[y]],
            p7 → member[e[x], domain[t]], p8 → member[pair[e[x], e[y]], t]}]] // Reverse

Out[4]= or[member[pair[e[x], e[y]], t], not[member[t, binhom[x, y]]],
          not[member[x, GROUPS]], not[member[y, GROUPS]]] == True

In[5]:= or[member[pair[e[x_], e[y_]], t_], not[member[t_, binhom[x_, y_]]],
          not[member[x_, GROUPS]], not[member[y_, GROUPS]]] := True
```

A binary homomorphism **t ∈ binhom[x, y]** satisfies the equation **t ∘ x = y ∘ (t ⊗ t)**. From this one can deduce an equation involving the one-sided inverse relation **image[inverse[x], {e[x]}]**. If one uses the **gp** wrapper, the one-sided inverse relation is automatically rewritten in terms of the (two-sided) inversion function **inv[x]** on one side of the equation.

Lemma. An equation involving the inversion function **inv[gp[y]]**.

```
In[6]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
          {u -> composite[t, gp[x]], v -> composite[gp[y], cross[t, t]],
           w -> composite[FIRST, id[cart[V, ids[gp[y]]]]]}] // Reverse

Out[6]= or[equal[composite[inverse[t], inv[gp[y]], t],
           image[inverse[gp[x]], image[inverse[t], set[e[gp[y]]]]]],
          not[equal[composite[t, gp[x]], composite[gp[y], cross[t, t]]]] == True

In[7]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. (Eliminating the equation involving the cross product.)

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
          not[implies[p1, p3]], {p1 → member[t, binhom[gp[x], gp[y]]],
           p2 -> equal[composite[t, gp[x]], composite[gp[y], cross[t, t]]],
           p3 -> equal[composite[inverse[t], inv[gp[y]], t],
             image[inverse[gp[x]], image[inverse[t], set[e[gp[y]]]]]]}]] // Reverse

Out[8]= or[equal[composite[inverse[t], inv[gp[y]], t],
           image[inverse[gp[x]], image[inverse[t], set[e[gp[y]]]]]],
          not[member[t, binhom[gp[x], gp[y]]]] == True

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

Lemma. (Eliminate both **gp** wrappers.)

```
In[10]:= Map[implies[and[member[x, GROUPS], member[y, GROUPS]], #] &,
          SubstTest[implies, and[equal[x, gp[u]], equal[y, gp[v]]],
            or[equal[composite[inverse[t], inv[y], t],
              image[inverse[x], image[inverse[t], set[e[y]]]]],
             not[member[t, binhom[x, y]]]], {u → x, v → y}] // Reverse]

Out[10]= or[equal[composite[inverse[t], inv[y], t],
            image[inverse[x], image[inverse[t], set[e[y]]]]], not[member[t, binhom[x, y]]],
           not[member[x, GROUPS]], not[member[y, GROUPS]]] == True

In[11]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. An inclusion that follows from the statement **pair[e[x], e[y]] ∈ t**.

```
In[12]:= SubstTest[implies, subclass[u, v],
           subclass[image[inverse[x], u], image[inverse[x], v]],
           {u -> set[e[x]], v -> image[inverse[t], set[e[y]]]}] // Reverse // MapNotNot

Out[12]= or[not[member[e[y], V]],
           not[member[pair[e[x], e[y]], t]], subclass[image[inverse[x], set[e[x]]],
            image[inverse[x], image[inverse[t], set[e[y]]]]] == True

In[13]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

The various lemmas can now be combined to derive the first of two monotonicity inclusions. The derivation is speeded up by omitting three proof steps as indicated by **(* ⋯ *)**.

Theorem. A monotonicity inclusion for binary homomorphisms between groups.

```
In[14]:= Map[not, SubstTest[and, (*implies[p1,p4],implies[p1,p5],*)
           implies[and[p4, p5], p6], implies[p1, p7], implies[p1, p8],
           (* implies[and[p6,p7,p8],p9],*) not[implies[p1, p9]],
           {p1 → and[member[x, GROUPS], member[y, GROUPS], member[t, binhom[x, y]]],
            p4 -> member[e[y], V], p5 -> member[pair[e[x], e[y]], t], p6 -> subclass[
              image[inverse[x], set[e[x]]], image[inverse[x], image[inverse[t], set[e[y]]]]],
            p7 → equal[inv[x], image[inverse[x], set[e[x]]]],
            p8 → equal[image[inverse[x], image[inverse[t], set[e[y]]]],
              composite[inverse[t], inv[y], t]],
            p9 -> subclass[inv[x], composite[inverse[t], inv[y], t]]}]] // Reverse

Out[14]= or[not[member[t, binhom[x, y]]], not[member[x, GROUPS]],
           not[member[y, GROUPS]], subclass[inv[x], composite[inverse[t], inv[y], t]]] == True

In[15]:= or[not[member[t_, binhom[x_, y_]]], not[member[x_, GROUPS]], not[member[y_, GROUPS]],
           subclass[inv[x_], composite[inverse[t_], inv[y_], t_]]] := True
```

Corollary. A second monotonicity statement for binary homomorphisms between groups.

```
In[16]:= Map[not, SubstTest[and, implies[p2, p3], implies[and[p1, p2], p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 -> and[member[x, GROUPS], member[y, GROUPS]], p2 -> member[t, binhom[x, y]],
            p3 → FUNCTION[t], p4 -> subclass[inv[x], composite[inverse[t], inv[y], t]],
            p5 -> subclass[composite[t, inv[x], inverse[t]], inv[y]]}]] // Reverse

Out[16]= or[not[member[t, binhom[x, y]]], not[member[x, GROUPS]],
           not[member[y, GROUPS]], subclass[composite[t, inv[x], inverse[t]], inv[y]]] == True

In[17]:= or[not[member[t_, binhom[x_, y_]]], not[member[x_, GROUPS]], not[member[y_, GROUPS]],
           subclass[composite[t_, inv[x_], inverse[t_]], inv[y_]]] := True
```

A more compact formulation is obtained by eliminating the variable **t**. This is done here using **reify** and **case**. A further trick used here is to take advantage of the following rewrite rule:

```
In[18]:= subclass[P[composite[Id, t]], monotone[x, y]]

Out[18]= subclass[composite[t, x, inverse[t]], y]
```

Corollary. If **x** and **y** are groups, then the set of binary homomorphisms from **x** to **y** is a subset of the set of relations that are monotone with respect to the inversion functions **inv[x]** and **inv[y]**.

```
In[19]:= Map[equal[V, domain[#]] &, SubstTest[reify, t,
           case[or[not[member[t, binhom[x, y]]], not[member[x, GROUPS]], not[member[y, GROUPS]],
             subclass[P[composite[Id, t]], w]]], w -> monotone[inv[x], inv[y]]]]

Out[19]= or[not[member[x, GROUPS]], not[member[y, GROUPS]],
           subclass[binhom[x, y], monotone[inv[x], inv[y]]]] == True

In[20]:= or[not[member[x_, GROUPS]], not[member[y_, GROUPS]],
           subclass[binhom[x_, y_], monotone[inv[x_], inv[y_]]]] := True
```

As amusing as these two monotonicity inclusions may be, equations are preferable. An intertwining equation will now be derived. To this end, the first step is to eliminate **inverse[t]** from the first of the two monotonicity inclusions.

Lemma. (Yet another inclusion, but one that involves only functions.)

```
In[21]:= Map[not, SubstTest[and, implies[p2, p3], implies[and[p1, p2], p4],
            implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
            {p1 -> and[member[x, GROUPS], member[y, GROUPS]], p2 -> member[t, binhom[x, y]],
             p3 → FUNCTION[t], p4 -> subclass[inv[x], composite[inverse[t], inv[y], t]],
             p5 -> subclass[composite[t, inv[x]], composite[inv[y], t]]}]] // Reverse

Out[21]= or[not[member[t, binhom[x, y]]], not[member[x, GROUPS]], not[member[y, GROUPS]],
           subclass[composite[t, inv[x]], composite[inv[y], t]]] == True

In[22]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. (Any subclass of a function is a restriction.)

```
In[24]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]],
           {u → composite[t, inv[x]], v → composite[inv[y], t]}] // Reverse

Out[24]= or[equal[composite[t, inv[x]], composite[inv[y], t, id[image[inv[x], domain[t]]]]],
           not[FUNCTION[composite[inv[y], t]]],
           not[subclass[composite[t, inv[x]], composite[inv[y], t]]]] == True

In[25]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. (Eliminating the **FUNCTION** hypotheses.)

```
In[36]:= Map[not, SubstTest[and, (*implies[p1,p2], *) implies[p1, p3], implies[p1, p4],
            (*implies[and[p3,p4],p5],*) implies[and[p2, p5], p6], not[implies[p1, p6]],
            {p1 -> and[member[x, GROUPS], member[y, GROUPS], member[t, binhom[x, y]]],
             p2 -> subclass[composite[t, inv[x]], composite[inv[y], t]],
             p3 → FUNCTION[inv[y]], p4 → FUNCTION[t],
             p5 -> FUNCTION[composite[inv[y], t]], p6 -> equal[composite[t, inv[x]],
               composite[inv[y], t, id[image[inv[x], domain[t]]]]]}]] // Reverse

Out[36]= or[equal[composite[t, inv[x]], composite[inv[y], t, id[image[inv[x], domain[t]]]]],
           not[member[t, binhom[x, y]]], not[member[x, GROUPS]], not[member[y, GROUPS]]] == True

In[37]:= (% /. {t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Main Theorem. A binary homomorphism between groups intertwines with the inversion functions of the groups.

```
In[48]:= Map[not, SubstTest[and, implies[p1, p2], (*implies[p1,p3],implies[p1,p4],*)
           implies[and[p3, p4], p5], (*implies[and[p2,p5],p6],*) not[implies[p1, p6]],
           {p1 -> and[member[x, GROUPS], member[y, GROUPS], member[t, binhom[x, y]]], p2 ->
             equal[composite[t, inv[x]], composite[inv[y], t, id[image[inv[x], domain[t]]]]],
            p3 → equal[domain[t], range[x]], p4 → equal[domain[inv[x]], range[x]],
            p5 → equal[image[inv[x], domain[t]], domain[t]],
            p6 -> equal[composite[t, inv[x]], composite[inv[y], t]]}]] // Reverse

Out[48]= or[equal[composite[t, inv[x]], composite[inv[y], t]],
          not[member[t, binhom[x, y]]], not[member[x, GROUPS]], not[member[y, GROUPS]]] == True

In[50]:= or[equal[composite[inv[y_], t_], composite[t_, inv[x_]]],
          not[member[t_, binhom[x_, y_]]],
          not[member[x_, GROUPS]], not[member[y_, GROUPS]]] := True
```