binary homomorphisms are ranges of subgroups

Johan G. F. Belinfante
2012 January 24

```
In[1]:= SetDirectory["l:"]; << goedel.12jan23a
```

```
:Package Title: goedel.12jan23a                2012 January 23 at 2:15 p.m.

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

It is now:  2012 Jan 24 at 10:32

Loading Simplification Rules

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

weightlimit = 40

Loading completed.

It is now:  2012 Jan 24 at 10:45
```

summary

In the **GOEDEL** program, the term **group** refers to what is commonly described as the composition law (or the multiplication table) of a group. The range of a group is the underlying set on which this acts. For example, the integer addition function **INTADD** is a group, and its range is the set **Z** of all integers. The subgroups of a group are determined by their ranges. For the case of **INTADD**, the range of a subgroup is the set of multiples of some integer **int[x]**.

```
In[2]:= member[image[INTDIV, set[int[x]]],
          image[IMAGE[SECOND], intersection[GROUPS, P[INTADD]]]]
```

```
Out[2]= True
```

The **direct product** of two groups **x** and **y** is the group **direct[x, y] = (x ⊗ y) ∘ TWIST**.

```
In[3]:= implies[and[member[x, GROUPS], member[y, GROUPS]], member[direct[x, y], GROUPS]]
```

```
Out[3]= True
```

In this notebook it is shown that a binary homomorphism from one group another is the range of a subgroup of their direct product.

derivation

The wrapper **gp[x]** is either a group or the empty set.

Theorem. A general result about **binhom** specialized to groups.

```
In[4]:= SubstTest[subclass, binhom[funpart[u], v],
          binclosed[direct[funpart[u], v]], {u → gp[x], v → gp[y]}] // Reverse

Out[4]= subclass[binhom[gp[x], gp[y]], binclosed[composite[cross[gp[x], gp[y]], TWIST]]] == True

In[5]:= subclass[binhom[gp[x_], gp[y_]],
          binclosed[composite[cross[gp[x_], gp[y_]], TWIST]]] := True
```

Corollary. (Restatement without the **gp** wrappers.)

```
In[6]:= SubstTest[implies, and[equal[x, gp[u]], equal[y, gp[v]]], subclass[binhom[x, y],
            binclosed[composite[cross[x, y], TWIST]]], {u → x, v → y}] // Reverse // MapNotNot

Out[6]= or[not[member[x, GROUPS]], not[member[y, GROUPS]],
          subclass[binhom[x, y], binclosed[composite[cross[x, y], TWIST]]]] == True

In[7]:= or[not[member[x_, GROUPS]], not[member[y_, GROUPS]],
          subclass[binhom[x_, y_], binclosed[composite[cross[x_, y_], TWIST]]]] := True
```

Binary homomorphisms between groups preserve inverses. It will be convenient to derive two versions of this statement with **gp** wrappers.

Lemma. Introducing **gp** wrappers. (This statement has two redundant literals.)

```
In[8]:= SubstTest[or, equal[composite[w, inv[u]], composite[inv[v], w]],
          not[member[w, binhom[u, v]]], not[member[u, GROUPS]],
          not[member[v, GROUPS]], {u → gp[x], v → gp[y]}] // Reverse

Out[8]= or[equal[0, gp[x]], equal[0, gp[y]],
          equal[composite[w, inv[gp[x]]], composite[inv[gp[y]], w]],
          not[member[w, binhom[gp[x], gp[y]]]]] == True

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

The redundant literals are easily eliminated.

Theorem. Binary homomorphisms of groups preserve inversion. (First version.)

```
In[10]:= SubstTest[and, implies[p, q], or[p, q], {p -> or[equal[0, gp[x]], equal[0, gp[y]]],
            q → or[equal[composite[w, inv[gp[x]]], composite[inv[gp[y]], w]],
              not[member[w, binhom[gp[x], gp[y]]]]]}] // MapNotNot

Out[10]= or[equal[composite[w, inv[gp[x]]], composite[inv[gp[y]], w]],
           not[member[w, binhom[gp[x], gp[y]]]]] == True

In[11]:= or[equal[composite[w_, inv[gp[x_]]], composite[inv[gp[y_]], w_]],
           not[member[w_, binhom[gp[x_], gp[y_]]]]] := True
```

A second version of the preservation of inversion will be derived below. The idea is just to put all of the inversions on one side of the equation. Because the general theory of binary homomorphisms applies to binary operations, many rewrite rules

involve the fixed point class of the domain of the binary operation. When specialized to groups, one can replace this with the range.

Lemma. A simplification rule.

```
In[12]:= SubstTest[implies, member[w, binhom[t, y]],
           equal[domain[w], fix[domain[t]]], t → gp[x]] // Reverse

Out[12]= or[equal[domain[w], range[gp[x]]], not[member[w, binhom[gp[x], y]]]] == True

In[13]:= or[equal[domain[w_], range[gp[x_]]], not[member[w_, binhom[gp[x_], y_]]]] := True
```

Theorem. A second form of the statement that binary homomorphisms for groups preserve inversion.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p1, p3], p4], implies[and[p2, p4], p5],
            not[implies[p1, p5]], {p1 -> member[w, binhom[gp[x], gp[y]]],
             p2 -> equal[composite[w, inv[gp[x]]], composite[inv[gp[y]], w]],
             p3 → equal[domain[w], range[gp[x]]],
             p4 → equal[w, composite[w, id[range[gp[x]]]]],
             p5 -> equal[w, composite[inv[gp[y]], w, inv[gp[x]]]]}]] // Reverse

Out[14]= or[equal[w, composite[inv[gp[y]], w, inv[gp[x]]]],
           not[member[w, binhom[gp[x], gp[y]]]]] == True

In[15]:= or[equal[w_, composite[inv[gp[y_]], w_, inv[gp[x_]]]],
           not[member[w_, binhom[gp[x_], gp[y_]]]]] := True
```

Corollary. (Eliminate the variable **w**.)

```
In[16]:= Map[equal[V, #] &,
          dif[binhom[gp[x], gp[y]], fix[IMAGE[inv[direct[gp[x], gp[y]]]]]] // complement //
           Normality]

Out[16]= subclass[binhom[gp[x], gp[y]], fix[IMAGE[cross[inv[gp[x]], inv[gp[y]]]]]] == True

In[17]:= subclass[binhom[gp[x_], gp[y_]], fix[IMAGE[cross[inv[gp[x_]], inv[gp[y_]]]]]] := True
```

The intersection of the collection of sets that are binary closed under a group and the collection of sets that are subvariant under inversion is the union of the set of ranges of subgroups of a group and the singleton of the empty set.

Theorem.

```
In[18]:= SubstTest[subclass, t, intersection[u, v],
           {t -> binhom[gp[x], gp[y]], u → binclosed[direct[gp[x], gp[y]]],
            v → fix[IMAGE[inv[direct[gp[x], gp[y]]]]]}] // Reverse

Out[18]= subclass[binhom[gp[x], gp[y]], union[image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]], set[0]]] == True

In[19]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary. (Eliminate **gp** wrappers. The union with **{0}** will be eliminated below.)

```
In[20]:= SubstTest[implies, and[equal[x, gp[u]], equal[y, gp[v]]], subclass[binhom[x, y], union[
             image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[x, y], TWIST]]]],
             set[0]]], {u → x, v → y}] // Reverse // MapNotNot

Out[20]= or[not[member[x, GROUPS]], not[member[y, GROUPS]],
           subclass[binhom[x, y], union[image[IMAGE[SECOND],
              intersection[GROUPS, P[composite[cross[x, y], TWIST]]]], set[0]]]] == True

In[21]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Technical Lemma. (This is needed to cope with a rewrite rule for the statement **0 ∈ binhom[x, y]**.)

```
In[22]:= SubstTest[implies, and[subclass[t, union[u, set[0]]], not[member[0, t]]],
           subclass[t, u], {t → binhom[x, y], u -> image[IMAGE[SECOND],
              intersection[GROUPS, P[composite[cross[x, y], TWIST]]]]}] // Reverse

Out[22]= or[equal[0, fix[domain[x]]], not[subclass[binhom[x, y], union[image[IMAGE[SECOND],
               intersection[GROUPS, P[composite[cross[x, y], TWIST]]], set[0]]],
           subclass[binhom[x, y], image[IMAGE[SECOND], intersection[GROUPS,
              P[composite[cross[x, y], TWIST]]]]]] == True

In[23]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Main Theorem. The set of binary homomorphisms from a group **x** to a group **y** is a subclass of the set of ranges of subgroups of the direct product of **x** and **y**.

```
In[24]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 → and[member[x, GROUPS], member[y, GROUPS]], p2 -> not[member[0, binhom[x, y]]],
             p3 → subclass[binhom[x, y], union[image[IMAGE[SECOND],
                 intersection[GROUPS, P[composite[cross[x, y], TWIST]]]], set[0]]],
             p4 → subclass[binhom[x, y], image[IMAGE[SECOND], intersection[GROUPS,
                 P[composite[cross[x, y], TWIST]]]]]}]] // Reverse

Out[24]= or[not[member[x, GROUPS]],
           not[member[y, GROUPS]], subclass[binhom[x, y], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[x, y], TWIST]]]]] == True

In[25]:= or[not[member[x_, GROUPS]], not[member[y_, GROUPS]],
           subclass[binhom[x_, y_], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[x_, y_], TWIST]]]]]] := True
```

Comment. The following special case was already available.

```
In[26]:= subclass[binhom[INTADD, INTADD],
          image[IMAGE[SECOND], intersection[GROUPS, P[direct[INTADD, INTADD]]]]]

Out[26]= True
```

This inclusion is not an equation. The function **inttimes[int[x]]** for multiplication by an integer **int[x]** is a binary homomorphism, and hence the range of a subgroup of **direct[INTADD, INTADD]**. Its inverse is also the range of a subgroup, but need not be a binary homomorphism. A simple example is the inverse of multiplication by the integer zero:

Theorem. The set **inverse[inttimes[plus[0]]]** is the range of a subgroup of **direct[INTADD, INTADD]**.

```
In[43]:= SubstTest[member, inverse[inttimes[t]], image[IMAGE[SECOND],
           intersection[GROUPS, P[direct[INTADD, INTADD]]]], t → plus[0]] // Reverse

Out[43]= member[cart[set[id[omega]], Z], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] == True

In[44]:= member[cart[set[id[omega]], Z], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] := True
```

Theorem. The set **inverse[inttimes[plus[0]]]** is not a binary homomorphism.

```
In[38]:= member[cart[set[id[omega]], Z], binhom[INTADD, INTADD]] // AssertTest

Out[38]= member[cart[set[id[omega]], Z], binhom[INTADD, INTADD]] == False

In[45]:= member[cart[set[id[omega]], Z], binhom[INTADD, INTADD]] := False
```