binary homomorphisms

Johan G. F. Belinfante
2009 February 5; corrected 2009 February 10

```
In[1]:= << l:goedel.09feb04a; << l:tools.m

:Package Title: goedel.09feb04a            2009 February 4 at 4:10 p.m.

It is now:  2009 Feb 10 at 11:0

Loading Simplification Rules

TOOLS.M                                    Revised 2009 February 4

weightlimit = 40
```

summary

By definition, a binary homomorphism **t ∈ binhom[x, y]** is a mapping from **fix[domain[x]]** to **fix[domain[y]]** which satisfies the equation $\mathbf{t \circ x = y \circ (t \otimes t)}$. In this notebook it is shown that if **x** and **y** are binary operations, this equation follows from the (seemingly weaker) inclusion $\mathbf{t \circ x \subset y \circ (t \otimes t)}$.

derivation

For simplicity, the derivation is first given using **funpart** and **binop** wrappers, which are subsequently eliminated.

Lemma. This derivation contains all the essential steps. Two temporary variables **u** and **v** are introduced, and eliminated from the final result. The key idea is that if $\mathbf{u \subset v}$ and **v** is a function, and if **domain[u] = domain[v]**, then **u = v**.

```
In[2]:= (Map[not, SubstTest[and, implies[p3, p4],
            implies[and[p1, p2, p4], p5], implies[p1, p6], implies[and[p1, p5, p6], p7],
            not[implies[p1, p7]], {p1 → and[equal[u, composite[funpart[h], binop[x]]],
               equal[v, composite[binop[y], cross[funpart[h], funpart[h]]]],
               member[funpart[h], map[fix[domain[binop[x]]], fix[domain[binop[y]]]]],
               subclass[u, v]], p2 → equal[domain[funpart[h]], fix[domain[binop[x]]]],
             p3 → subclass[range[funpart[h]], fix[domain[binop[y]]]], p4 →
              equal[domain[funpart[h]], image[inverse[funpart[h]], fix[domain[binop[y]]]]],
             p5 → equal[domain[u], domain[v]], p6 → FUNCTION[v], p7 → equal[u, v]}]] //
         Reverse) /. {u -> composite[funpart[h], binop[x]],
        v -> composite[binop[y], cross[funpart[h], funpart[h]]]}

Out[2]= or[equal[composite[binop[y], cross[funpart[h], funpart[h]]],
           composite[funpart[h], binop[x]]],
         not[member[funpart[h], map[fix[domain[binop[x]]], fix[domain[binop[y]]]]]],
         not[subclass[composite[funpart[h], binop[x]],
           composite[binop[y], cross[funpart[h], funpart[h]]]]]] == True

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

Theorem. Here the **funpart** wrapper is removed, and the constructor **binhom** is introduced.

```
In[4]:= (Map[not, SubstTest[and, implies[and[p0, p1], p2], implies[and[p1, p2], p3],
            implies[and[p1, p3], p4], not[implies[and[p0, p1], p4]], {p0 → equal[h, t],
             p1 → and[member[t, map[fix[domain[binop[x]]], fix[domain[binop[y]]]]],
               subclass[composite[t, binop[x]], composite[binop[y], cross[t, t]]]],
             p2 → equal[t, funpart[h]], p3 -> equal[composite[binop[y], cross[t, t]],
               composite[t, binop[x]]],
             p4 → member[t, binhom[binop[x], binop[y]]]}]] // Reverse) /. h → t

Out[4]= or[member[t, binhom[binop[x], binop[y]]],
         not[member[t, map[fix[domain[binop[x]]], fix[domain[binop[y]]]]]],
         not[subclass[composite[t, binop[x]], composite[binop[y], cross[t, t]]]]] == True

In[5]:= or[member[t_, binhom[binop[x_], binop[y_]]],
         not[member[t_, map[fix[domain[binop[x_]]], fix[domain[binop[y_]]]]]],
         not[subclass[composite[t_, binop[x_]], composite[binop[y_], cross[t_, t_]]]]] := True
```

Corollary. If **x** and **y** are binary operations and if the mapping **t: fix[domain[x]] → fix[domain[y]]** satisfies the condition **t ∘ x ⊂ y ∘ (t ⊗ t)**, then **t** is a binary homomorphism: **t ∈ binhom[x, y]**. (The corollary follows from the theorem by removing the **binop** wrappers in a standard fashion.)

```
In[6]:= SubstTest[implies, and[equal[x, binop[u]],
           equal[y, binop[v]], member[t, map[fix[domain[x]], fix[domain[y]]]],
           subclass[composite[t, x], composite[y, cross[t, t]]]],
          member[t, binhom[x, y]], {u → x, v → y}] // Reverse

Out[6]= or[member[t, binhom[x, y]], not[member[t, map[fix[domain[x]], fix[domain[y]]]]],
          not[member[x, BINOPS]], not[member[y, BINOPS]],
          not[subclass[composite[t, x], composite[y, cross[t, t]]]]] == True

In[7]:= or[member[t_, binhom[x_, y_]], not[member[t_, map[fix[domain[x_]], fix[domain[y_]]]]],
          not[member[x_, BINOPS]], not[member[y_, BINOPS]],
          not[subclass[composite[t_, x_], composite[y_, cross[t_, t_]]]]] := True
```