binary homomorphism for semigroups

Johan G. F. Belinfante
2006 September 28

```
In[1]:= SetDirectory["l:"]; << goedel85.27a; << tools.m

:Package Title: goedel85.27a          2006 September 27 at 9:40 p.m.

It is now:  2006 Sep 28 at 8:32

Loading Simplification Rules

TOOLS.M                     Revised 2006 September 24

weightlimit = 40
```

summary

The image of a semigroup under a binary homomorphism is a semigroup. The derivation of this theorem is expedited by not supplying all steps of the proofs, relying on rewrite rules to supply missing steps.

minimal derivation of first step

The first step is to form the composite of the equation for associativity with **w** on the left and twice replace **composite[w,x]** with **composite[y, cross[w,w]]**. For the latter, a lemma is useful:

```
In[2]:= SubstTest[implies, and[equal[s, t], equal[composite[s, u], composite[s, v]]],
          equal[composite[t, u], composite[t, v]]],
          {s → composite[w, x], t → composite[y, cross[w, w]],
           u → cross[x, Id], v → composite[cross[Id, x], ASSOC]}]

Out[2]= or[equal[composite[y, cross[composite[w, x], w]],
            composite[y, cross[w, composite[w, x]], ASSOC]],
           not[equal[composite[w, x], composite[y, cross[w, w]]]],
           not[equal[composite[w, x, cross[x, Id]], composite[w, x, cross[Id, x], ASSOC]]]] == True

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

Theorem.

```
In[4]:= Map[not, SubstTest[and, implies[and[p2, p3], p4], not[implies[and[p1, p2], p5]],
          {p1 → equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
           p2 → equal[composite[w, x], composite[y, cross[w, w]]],
           p3 -> equal[composite[w, x, cross[x, Id]], composite[w, x, cross[Id, x], ASSOC]],
           p4 -> equal[composite[y, cross[composite[w, x], w]],
             composite[y, cross[w, composite[w, x]], ASSOC]],
           p5 → equal[composite[y, cross[composite[y, cross[w, w]], w]],
             composite[y, cross[w, composite[y, cross[w, w]]], ASSOC]]}]]

Out[4]= or[equal[composite[y, cross[composite[y, cross[w, w]], w]],
           composite[y, cross[w, composite[y, cross[w, w]]], ASSOC]],
          not[equal[composite[w, x], composite[y, cross[w, w]]]],
          not[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]] == True

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

step two

The second step is to form the composite on the right with **inverse[cross[cross[w,w],w]]** and replace **w** with **funpart[z]**.

```
In[6]:= SubstTest[implies, equal[s, t], equal[composite[s, u], composite[t, u]],
          {s -> composite[y, cross[composite[y, cross[w, w]], w]],
           t -> composite[y, cross[w, composite[y, cross[w, w]]], ASSOC],
           u -> inverse[cross[cross[w, w], w]]}] /. w → funpart[z]

Out[6]= or[equal[
           composite[y, cross[composite[y, id[cart[range[funpart[z]], range[funpart[z]]]]],
             id[range[funpart[z]]]]], composite[y, cross[id[range[funpart[z]]],
             composite[y, id[cart[range[funpart[z]], range[funpart[z]]]]]]], ASSOC]], not[
           equal[composite[y, cross[composite[y, cross[funpart[z], funpart[z]]], funpart[z]]],
            composite[y, cross[funpart[z], composite[y, cross[funpart[z], funpart[z]]]]],
             ASSOC]]]] == True

In[7]:= (% /. {y → y_, z → z_}) /. Equal → SetDelayed
```

Removing the **funpart** wrappers yields:

```
In[8]:= SubstTest[implies,
         and[equal[w, funpart[z]], equal[composite[y, cross[composite[y, cross[w, w]], w]],
           composite[y, cross[w, composite[y, cross[w, w]]], ASSOC]]], equal[composite[y,
           cross[composite[y, id[cart[range[w], range[w]]]], id[range[w]]]], composite[y,
           cross[id[range[w]], composite[y, id[cart[range[w], range[w]]]]], ASSOC]], z → w]

Out[8]= or[equal[composite[y, cross[composite[y, id[cart[range[w], range[w]]]], id[range[w]]]],
           composite[y, cross[id[range[w]], composite[y, id[cart[range[w], range[w]]]]]],
            ASSOC]], not[equal[composite[y, cross[composite[y, cross[w, w]], w]],
            composite[y, cross[w, composite[y, cross[w, w]]]], ASSOC]]], not[FUNCTION[w]]] == True

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

main theorem

From the definition of **associative** one finds:

```
In[10]:= SubstTest[implies, and[subclass[x, cart[cart[V, V], V]],
            equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]],
          associative[x], x → composite[y, id[cart[r, r]]]]

Out[10]= or[associative[composite[y, id[cart[r, r]]]],
            not[equal[composite[y, cross[composite[id[r], y, id[cart[r, r]]], id[r]]],
              composite[y, cross[id[r], composite[id[r], y, id[cart[r, r]]]], ASSOC]]]] == True

In[11]:= (% /. {r → r_, y → y_}) /. Equal → SetDelayed
```

The following lemma connects this with the result derived in the preceding direction:

```
In[12]:= SubstTest[implies, and[equal[u, v],
            equal[composite[y, cross[u, id[r]]], composite[y, cross[id[r], u], ASSOC]]],
          equal[composite[y, cross[v, id[r]]], composite[y, cross[id[r], v], ASSOC]],
          {u → composite[y, id[cart[r, r]]], v -> composite[id[r], y, id[cart[r, r]]]}]

Out[12]= or[equal[composite[y, cross[composite[id[r], y, id[cart[r, r]]], id[r]]],
             composite[y, cross[id[r], composite[id[r], y, id[cart[r, r]]]], ASSOC]],
            not[equal[composite[y, cross[composite[y, id[cart[r, r]]], id[r]]],
              composite[y, cross[id[r], composite[y, id[cart[r, r]]]], ASSOC]]],
            not[subclass[image[y, cart[r, r]], r]]] == True

In[13]:= (% /. {r → r_, y → y_}) /. Equal → SetDelayed
```

It remains to assemble all these results. The following derivation deliberately omits three steps to reduce execution time: **implies[p2,p4]**, **implies[and[p3,p5],p7]**, and **implies[and[p6,p8],p9]**. This yields the main theorem: associativity is preserved by binary homomorphisms.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p5],
            implies[p2, p6], implies[and[p4, p7], p8], implies[p9, p10],
            not[implies[and[p1, p2], p10]], {p1 → associative[x], p2 → member[w, binhom[x, y]],
             p3 -> equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
             p4 → FUNCTION[w], p5 -> equal[composite[w, x], composite[y, cross[w, w]]],
             p6 -> subclass[image[y, cart[range[w], range[w]]], range[w]],
             p7 -> or[equal[composite[y, cross[composite[y, cross[w, w]], w]],
                composite[y, cross[w, composite[y, cross[w, w]]], ASSOC]]], p8 -> equal[composite[
                y, cross[composite[y, id[cart[range[w], range[w]]]], id[range[w]]]], composite[y,
                cross[id[range[w]], composite[y, id[cart[range[w], range[w]]]]], ASSOC]], p9 ->
              equal[composite[y, cross[composite[id[range[w]], y, id[cart[range[w], range[w]]]],
                 id[range[w]]]], composite[y, cross[id[range[w]],
                 composite[id[range[w]], y, id[cart[range[w], range[w]]]]], ASSOC]],
             p10 → associative[composite[y, id[cart[range[w], range[w]]]]]}]]

Out[14]= or[associative[composite[y, id[cart[range[w], range[w]]]]],
            not[associative[x]], not[member[w, binhom[x, y]]]] == True
```

```
In[15]:= or[associative[composite[y_, id[cart[range[w_], range[w_]]]]],
           not[associative[x_]], not[member[w_, binhom[x_, y_]]]] := True
```

Corollary. The image of a semigroup under a binary homomorphism is a semigroup.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p3], implies[p1, p4],
           implies[and[p2, p3], p5], implies[and[p2, p4], p6], implies[and[p5, p6], p7],
           not[implies[and[p1, p2], p7]], {p1 → member[x, SEMIGPS],
            p2 → member[w, binhom[x, y]], p3 → member[x, BINOPS], p4 → associative[x],
            p5 → member[composite[y, id[cart[range[w], range[w]]]], BINOPS],
            p6 -> associative[composite[y, id[cart[range[w], range[w]]]]],
            p7 → member[composite[y, id[cart[range[w], range[w]]]], SEMIGPS]}]]

Out[16]= or[member[composite[y, id[cart[range[w], range[w]]]], SEMIGPS],
           not[member[w, binhom[x, y]]], not[member[x, SEMIGPS]]] == True

In[17]:= or[member[composite[y_, id[cart[range[w_], range[w_]]]], SEMIGPS],
           not[member[w_, binhom[x_, y_]]], not[member[x_, SEMIGPS]]] := True
```