binary homomorphisms, part 3

Johan G. F. Belinfante
2006 July 18

```
In[1]:= SetDirectory["l:"]; << goedel83.18b; << tools.m

        :Package Title: goedel83.18b         2006 July 18 at 9:35 p.m.

        It is now:  2006 Jul 20 at 16:27

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 July 18

        weightlimit = 40
```

summary

In this third part of a series of notebooks on binary homomorphisms, it is shown that the set of endomorphisms of **NATADD** is equal to the set of all functions of the form **times[x]** for some natural number **x**.

lemmas

In this section some convenient lemmas will be derived that make it possible to shorten the proofs of theorems about binary homomorphisms. Since **binhom[x,y]** is a subclass of **map[fix[domain[x]],fix[domain[y]]]**, the basic facts about mappings hold for binary homomorphisms. By showing this once and for all, it will not be necessary to repeat such arguments over and over again. By the transitivity of inclusion, one has:

```
In[2]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → binhom[x, y], v → map[fix[domain[x]], fix[domain[y]]], w → FUNS}]

Out[2]= subclass[binhom[x, y], FUNS] == True

In[3]:= subclass[binhom[x_, y_], FUNS] := True
```

Corollary.

```
In[4]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → binhom[x, y], v → FUNS, w → P[cart[V, V]]}]

Out[4]= subclass[U[binhom[x, y]], cart[V, V]] == True

In[5]:= % /. Equal → SetDelayed
```

The following rewrite rule holds:

```
In[6]:= equal[intersection[P[cart[V, V]], binhom[x, y]], binhom[x, y]]

Out[6]= True

In[7]:= intersection[binhom[x_, y_], P[cart[V, V]]] := binhom[x, y]
```

Corollary.

```
In[8]:= SubstTest[implies, and[member[w, u], subclass[u, v]],
         member[w, v], {u -> binhom[x, y], v → FUNS}]

Out[8]= or[FUNCTION[w], not[member[w, binhom[x, y]]]] == True

In[9]:= or[FUNCTION[w_], not[member[w_, binhom[x_, y_]]]] := True

In[10]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[w, binhom[x, y]], p2 → FUNCTION[w], p3 → subclass[w, cart[V, V]]}]]

Out[10]= or[not[member[w, binhom[x, y]]], subclass[w, cart[V, V]]] == True

In[11]:= or[not[member[w_, binhom[x_, y_]]], subclass[w, cart[V, V]]] := True
```

For the domain of a binary homomorphism, one has an equation:

```
In[12]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[w, binhom[x, y]], p2 → member[w, map[fix[domain[x]], fix[domain[y]]]],
            p3 → equal[domain[w], fix[domain[x]]]}]]

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

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

Similarly, for the range of a binary homomorphism, one has an inclusion:

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 → member[w, binhom[x, y]], p2 → member[w, map[fix[domain[x]], fix[domain[y]]]],
            p3 → subclass[range[w], fix[domain[y]]]}]]

Out[14]= or[not[member[w, binhom[x, y]]], subclass[range[w], fix[domain[y]]]] == True

In[15]:= or[not[member[w_, binhom[x_, y_]]], subclass[range[w_], fix[domain[y_]]]] := True
```

an inclusion obtained using reify

Lemma.

```
In[16]:= Map[equal[0, #] &, ImageComp[id[complement[set[0]]],
            VERTSECT[inverse[rotate[NATMUL]]], complement[omega]]] // Reverse

Out[16]= subclass[image[VERTSECT[inverse[rotate[NATMUL]]], complement[omega]], set[0]] == True

In[17]:= % /. Equal → SetDelayed
```

Theorem. The inclusion in the lemma can be sharpened to an equation, and made into a rewrite rule.

```
In[18]:= equal[image[VERTSECT[inverse[rotate[NATMUL]]], complement[omega]], set[0]]

Out[18]= True

In[19]:= image[VERTSECT[inverse[rotate[NATMUL]]], complement[omega]] := set[0]
```

Recall that **times[x]** is an endomorphism of **NATADD** for any natural number **x**. Wrapping **x** with **nat**, and applying **reify**, the variable **x** can be eliminated, leading to this variable-free restatement in the form of an inclusion.

```
In[20]:= Map[equal[0, #] &,
           SubstTest[reify, x, dif[set[times[nat[x]]], z], z → binhom[NATADD, NATADD]]] // Reverse

Out[20]= subclass[image[IMAGE[SWAP], image[VERTSECT[inverse[rotate[NATMUL]]], omega]],
           binhom[NATADD, NATADD]] == True

In[21]:= % /. Equal → SetDelayed
```

It will be shown below that conversely every endomorphism of **NATADD** is of the form **times[x]**, and therefore this inclusion can be replaced with an equation. It is convenient to move the **IMAGE[SWAP]** to the other side of the inclusion. To do so, some lemmas are needed:

```
In[22]:= ImageComp[IMAGE[SWAP], id[P[cart[V, V]]], binhom[x, y]] // Reverse

Out[22]= image[IMAGE[SWAP], binhom[x, y]] == image[INVERSE, binhom[x, y]]

In[23]:= image[IMAGE[SWAP], binhom[x_, y_]] := image[INVERSE, binhom[x, y]]

In[24]:= SubstTest[VERTSECT,
           composite[id[x], inverse[rotate[NATMUL]]], x → cart[V, V]] // Reverse

Out[24]= composite[IMAGE[id[cart[V, V]]], VERTSECT[inverse[rotate[NATMUL]]]] ==
          VERTSECT[inverse[rotate[NATMUL]]]

In[25]:= composite[IMAGE[id[cart[V, V]]], VERTSECT[inverse[rotate[NATMUL]]]] :=
          VERTSECT[inverse[rotate[NATMUL]]]
```

Lemma.

```
In[26]:= ImageComp[IMAGE[id[cart[V, V]]], VERTSECT[inverse[rotate[NATMUL]]], x] // Reverse

Out[26]= image[IMAGE[id[cart[V, V]]], image[VERTSECT[inverse[rotate[NATMUL]]], x]] ==
          image[VERTSECT[inverse[rotate[NATMUL]]], x]

In[27]:= image[IMAGE[id[cart[V, V]]], image[VERTSECT[inverse[rotate[NATMUL]]], x_]] :=
          image[VERTSECT[inverse[rotate[NATMUL]]], x]
```

Corollary.

```
In[28]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> image[IMAGE[SWAP], image[VERTSECT[inverse[rotate[NATMUL]]], omega]],
           v -> binhom[NATADD, NATADD], w → IMAGE[SWAP]}]

Out[28]= subclass[image[VERTSECT[inverse[rotate[NATMUL]]], omega],
           image[INVERSE, binhom[NATADD, NATADD]]] == True

In[29]:= % /. Equal → SetDelayed
```

---

endomorphisms of NATADD send 0 to 0.

Lemma.

```
In[30]:= SubstTest[implies, equal[u, v],
           equal[APPLY[u, w], APPLY[v, w]], {u -> composite[x, NATADD],
            v -> composite[NATADD, cross[x, x]], w → PAIR[0, 0]}] /. x → funpart[z]

Out[30]= or[equal[0, APPLY[funpart[z], 0]],
           not[equal[composite[NATADD, cross[funpart[z], funpart[z]]],
             composite[funpart[z], NATADD]]], not[member[0, domain[funpart[z]]]]] == True

In[31]:= (% /. z → z_) /. Equal → SetDelayed
```

Removing the **funpart** wrapper yields:

```
In[32]:= SubstTest[implies, equal[x, funpart[z]], or[equal[0, APPLY[x, 0]],
           not[equal[composite[NATADD, cross[x, x]], composite[x, NATADD]]],
           not[member[0, domain[x]]]], z → x]

Out[32]= or[equal[0, APPLY[x, 0]],
           not[equal[composite[NATADD, cross[x, x]], composite[x, NATADD]]],
           not[FUNCTION[x]], not[member[0, domain[x]]]] == True

In[33]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Every endomorphism of **NATADD** takes **0** to **0**.

```
In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
           implies[p2, p5], implies[p1, p6], implies[and[p4, p5, p6], p7], not[implies[p1, p7]],
           {p1 → member[x, binhom[NATADD, NATADD]], p2 → member[x, map[omega, omega]],
            p3 → equal[omega, domain[x]], p4 → member[0, domain[x]], p5 → FUNCTION[x],
            p6 -> equal[composite[NATADD, cross[x, x]], composite[x, NATADD]],
            p7 -> equal[0, APPLY[x, 0]]}]]

Out[34]= or[equal[0, APPLY[x, 0]], not[member[x, binhom[NATADD, NATADD]]]] == True

In[35]:= or[equal[0, APPLY[x_, 0]], not[member[x_, binhom[NATADD, NATADD]]]] := True
```

a recursion relation satisfied by endomorphisms of NATADD

Lemma.

```
In[36]:= SubstTest[implies, equal[u, v],
           equal[composite[u, w], composite[v, w]], {u -> composite[x, NATADD],
            v -> composite[NATADD, cross[x, x]], w → LEFT[set[0]]}] /. x → funpart[z]

Out[36]= or[equal[composite[plus[APPLY[funpart[z], set[0]]], funpart[z]],
            composite[funpart[z], id[omega], SUCC]],
           not[equal[composite[NATADD, cross[funpart[z], funpart[z]]],
             composite[funpart[z], NATADD]]]] == True

In[37]:= (% /. z → z_) /. Equal → SetDelayed
```

Removing the **funpart** wrapper yields:

```
In[38]:= SubstTest[implies, equal[x, funpart[z]],
          or[equal[composite[plus[APPLY[x, set[0]]], x], composite[x, id[omega], SUCC]],
           not[equal[composite[NATADD, cross[x, x]], composite[x, NATADD]]]], z → x]

Out[38]= or[equal[composite[plus[APPLY[x, set[0]]], x], composite[x, id[omega], SUCC]],
           not[equal[composite[NATADD, cross[x, x]], composite[x, NATADD]]],
           not[FUNCTION[x]]] == True

In[39]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Every endomorphism of **NATADD** satisfies a certain recursion relation:

```
In[40]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p1, p4],
           implies[p4, p5], implies[and[p2, p3], p6], implies[and[p5, p6], p7],
           not[implies[p1, p7]], {p1 → member[x, binhom[NATADD, NATADD]], p2 → FUNCTION[x],
            p3 -> equal[composite[NATADD, cross[x, x]], composite[x, NATADD]],
            p4 → member[x, map[omega, omega]], p5 → equal[domain[x], omega],
            p6 -> equal[composite[plus[APPLY[x, set[0]]], x], composite[x, id[omega], SUCC]],
            p7 -> equal[composite[plus[APPLY[x, set[0]]], x], composite[x, SUCC]]}]]

Out[40]= or[equal[composite[x, SUCC], composite[plus[APPLY[x, set[0]]], x]],
           not[member[x, binhom[NATADD, NATADD]]]] == True

In[41]:= (% /. x → x_) /. Equal → SetDelayed
```

applying uniqueness of iterate

The uniqueness theorem for **iterate** implies:

```
In[42]:= SubstTest[implies, and[equal[composite[u, w], composite[w, SUCC]],
                    equal[image[w, set[0]], v]],
                  equal[composite[w, id[omega]], iterate[u, v]],
          {u → plus[nat[x]], v → set[0], w → funpart[z]}]

Out[42]= or[equal[composite[funpart[z], id[omega]], times[nat[x]]],
           not[equal[0, APPLY[funpart[z], 0]]],
           not[equal[composite[funpart[z], SUCC], composite[plus[nat[x]], funpart[z]]]]] == True

In[43]:= (% /. {x → x_, z → z_}) /. Equal → SetDelayed
```

Removing the **funpart** and **nat** wrappers yields:

```
In[44]:= SubstTest[implies, and[equal[x, funpart[z]], equal[y, nat[t]]],
          or[equal[composite[x, id[omega]], times[y]], not[equal[0, APPLY[x, 0]]],
           not[equal[composite[x, SUCC], composite[plus[y], x]]]], {t → y, z → x}]

Out[44]= or[equal[composite[x, id[omega]], times[y]], not[equal[0, APPLY[x, 0]]],
           not[equal[composite[x, SUCC], composite[plus[y], x]]],
           not[FUNCTION[x]], not[member[y, omega]]] == True

In[45]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[46]:= SubstTest[implies, and[member[w, map[u, v]], member[x, u]],
          member[APPLY[w, x], v], {u → omega, v → omega, x → set[0]}]

Out[46]= or[member[APPLY[w, set[0]], omega], not[member[w, map[omega, omega]]]] == True

In[47]:= (% /. w → w_) /. Equal → SetDelayed
```

Putting together all these results, one finds the following, which will need to be cleaned up a bit:

```
In[48]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p4], implies[p2, p3], implies[p1, p5], implies[p1, p6],
           implies[p2, p7], implies[p1, p8], implies[and[p5, p6, p7, p8], p9],
           not[implies[p1, p9]], {p1 → member[x, binhom[NATADD, NATADD]],
            p2 → member[x, map[omega, omega]], p3 → member[APPLY[x, set[0]], omega],
            p4 → equal[composite[x, NATADD], composite[NATADD, cross[x, x]]], p5 → FUNCTION[x],
            p6 -> equal[composite[x, SUCC], composite[plus[APPLY[x, set[0]]], x]],
            p7 → member[APPLY[x, set[0]], omega], p8 -> equal[0, APPLY[x, 0]],
            p9 -> equal[composite[x, id[omega]], times[APPLY[x, set[0]]]]}]]

Out[48]= or[equal[composite[x, id[omega]], times[APPLY[x, set[0]]]],
           not[member[x, binhom[NATADD, NATADD]]]] == True

In[49]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[50]:= SubstTest[implies, member[x, binhom[w, y]],
          equal[domain[x], fix[domain[w]]], w → NATADD]

Out[50]= or[equal[omega, domain[x]], not[member[x, binhom[NATADD, y]]]] == True

In[51]:= or[equal[omega, domain[x_]], not[member[x_, binhom[NATADD, y_]]]] := True
```

Theorem. Every endomorphism of **NATADD** is a function of the form **times[z]**.

```
In[52]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p1, p4], implies[and[p2, p3, p4], p5], not[implies[p1, p5]],
           {p1 -> member[x, binhom[NATADD, NATADD]], p2 → equal[domain[x], omega],
            p3 -> equal[composite[x, id[omega]], times[APPLY[x, set[0]]]],
            p4 → FUNCTION[x], p5 → equal[x, times[APPLY[x, set[0]]]]}]]

Out[52]= or[equal[x, times[APPLY[x, set[0]]]], not[member[x, binhom[NATADD, NATADD]]]] == True

In[53]:= or[equal[x_, times[APPLY[x_, set[0]]]], not[member[x_, binhom[NATADD, NATADD]]]] := True
```

---

eliminating the variable x

The first step is to derive this membership rule:

```
In[54]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[x], v → omega, w → composite[VERTSECT[inverse[rotate[NATMUL]]]]}]

Out[54]= or[member[inverse[times[x]], image[VERTSECT[inverse[rotate[NATMUL]]], omega]],
           not[member[x, omega]]] == True

In[55]:= or[member[inverse[times[x_]], image[VERTSECT[inverse[rotate[NATMUL]]], omega]],
           not[member[x_, omega]]] := True

In[56]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[and[p2, p4], p5], not[implies[p1, p5]],
           {p1 → member[x, binhom[NATADD, NATADD]], p2 -> equal[x, times[APPLY[x, set[0]]]],
            p3 → member[x, map[omega, omega]], p4 → member[APPLY[x, set[0]], omega],
            p5 → member[inverse[x], image[VERTSECT[inverse[rotate[NATMUL]]], omega]]}]]

Out[56]= or[member[inverse[x], image[VERTSECT[inverse[rotate[NATMUL]]], omega]],
           not[member[x, binhom[NATADD, NATADD]]]] == True

In[57]:= (% /. x → x_) /. Equal → SetDelayed

In[58]:= Map[equal[V, #] &,
          union[image[inverse[IMAGE[SWAP]], image[VERTSECT[inverse[rotate[NATMUL]]], omega]],
           complement[binhom[NATADD, NATADD]]] // Normality]

Out[58]= subclass[image[INVERSE, binhom[NATADD, NATADD]],
           image[VERTSECT[inverse[rotate[NATMUL]]], omega]] == True

In[59]:= % /. Equal → SetDelayed
```

```
In[60]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[INVERSE, binhom[NATADD, NATADD]],
           v -> image[VERTSECT[inverse[rotate[NATMUL]]], omega]}]

Out[60]= True == equal[image[INVERSE, binhom[NATADD, NATADD]],
           image[VERTSECT[inverse[rotate[NATMUL]]], omega]]

In[61]:= image[VERTSECT[inverse[rotate[NATMUL]]], omega] :=
          image[INVERSE, binhom[NATADD, NATADD]]
```

Comment. It is interesting to note that a rather similar equation has been derived previously for the expression obtained when one replaces **NATMUL** with **NATADD**.

```
In[62]:= image[VERTSECT[inverse[rotate[NATADD]]], omega]

Out[62]= image[INVERSE, range[PLUS]]
```