MIXTIMES

Johan G. F. Belinfante
2006 December 5

```
In[1]:= SetDirectory["l:"]; << goedel88.03a; << tools.m

        :Package Title: goedel88.03a          2006 December 3 at 6:50 a.m.

        It is now:  2006 Dec 5 at 15:9

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 November 22

        weightlimit = 40
```

summary

In this notebook, the binary homomorphisms from **NATADD** to **INTADD** are studied. Each such homomorphisms correspond to mixed multiplication of natural numbers with a fixed integer yielding integers. There is a one-to-one correspondence **MIXTIMES** between integers and members of **binhom[NATADD, INTADD]**. A **class**-wrapped membership rule for this function has been introduced:

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= FirstMatch[class[t_, member[w_, HoldPattern[MIXTIMES]]]]

Out[3]= class[t_, member[w_, MIXTIMES]] := ReleaseHold[Module[{u = Unique[],
          v = Unique[]}, class[t, exists[u, v, and[equal[w, pair[u, v]], equal[
          image[v, set[set[0]]], set[u]], member[v, binhom[NATADD, INTADD]]]]]]]
```

This one-to-one function **MIXTIMES** takes each integer **z** to the unique binary homomorphism which maps the natural number **1 = set[0]** to the specified integer **z**.

normalization

Theorem.

```
In[4]:= Map[not,
          SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4], not[implies[p1, p4]],
           {p1 → member[x, binhom[NATADD, INTADD]], p2 -> member[x, map[omega, Z]],
            p3 → equal[domain[x], omega], p4 → member[set[0], domain[x]]}]] // Reverse

Out[4]= or[member[set[0], domain[x]], not[member[x, binhom[NATADD, INTADD]]]] == True
```

```
In[5]:= or[member[set[0], domain[x_]], not[member[x_, binhom[NATADD, INTADD]]]] := True
```

The following variable-free corollaries are needed in the next section:

```
In[6]:= Map[equal[V, #] &, SubstTest[class, x,
          or[member[set[0], domain[x]], not[member[x, y]]], y -> binhom[NATADD, INTADD]]]

Out[6]= equal[0, intersection[binhom[NATADD, INTADD], P[complement[cart[set[set[0]], V]]]]] ==
         True

In[7]:= intersection[binhom[NATADD, INTADD], P[complement[cart[set[set[0]], V]]]] := 0
```

Corollary. (Needed to normalize **MIXTIMES**.)

```
In[8]:= equal[intersection[binhom[NATADD, INTADD],
          complement[P[complement[cart[set[set[0]], V]]]], binhom[NATADD, INTADD]]

Out[8]= True

In[9]:= intersection[binhom[NATADD, INTADD],
          complement[P[complement[cart[set[set[0]], V]]]] := binhom[NATADD, INTADD]
```

Theorem.

```
In[10]:= MIXTIMES // Normality // Reverse

Out[10]= composite[id[binhom[NATADD, INTADD]], inverse[eval[set[0]]]] == MIXTIMES

In[11]:= composite[id[binhom[NATADD, INTADD]], inverse[eval[set[0]]]] := MIXTIMES
```

---

some properties of MIXTIMES

Corollary.

```
In[12]:= Assoc[Id, id[binhom[NATADD, INTADD]], inverse[eval[set[0]]]]

Out[12]= composite[Id, MIXTIMES] == MIXTIMES

In[13]:= composite[Id, MIXTIMES] := MIXTIMES

In[14]:= composite[eval[set[0]], id[binhom[NATADD, INTADD]]] // DoubleInverse

Out[14]= composite[eval[set[0]], id[binhom[NATADD, INTADD]]] == inverse[MIXTIMES]

In[15]:= composite[eval[set[0]], id[binhom[NATADD, INTADD]]] := inverse[MIXTIMES]
```

Corollary.

```
In[16]:= SubstTest[FUNCTION, composite[eval[set[0]], id[x]],
           x → binhom[NATADD, INTADD]] // Reverse

Out[16]= FUNCTION[inverse[MIXTIMES]] == True
```

In[17]:=
```
FUNCTION[inverse[MIXTIMES]] := True
```

domain

In[18]:=
```
SubstTest[range, composite[eval[set[0]], id[x]], x → binhom[NATADD, INTADD]] // Reverse
```

Out[18]=
```
domain[MIXTIMES] == Z
```

In[19]:=
```
domain[MIXTIMES] := Z
```

range

In[20]:=
```
Map[not, SubstTest[and, implies[p1, p3], implies[p1, p4],
   implies[and[p2, p3, p4], p5], not[implies[and[p1, p2], p5]],
   {p1 -> member[w, map[x, y]], p2 -> member[t, x], p3 → FUNCTION[w],
    p4 → equal[x, domain[w]], p5 -> member[t, domain[funpart[w]]]}]] // Reverse
```

Out[20]=
```
or[member[t, domain[funpart[w]]], not[member[t, x]], not[member[w, map[x, y]]]] == True
```

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

In[22]:=
```
Map[equal[V, #] &,
 SubstTest[class, t, implies[and[member[w, x], member[t, u]], member[t, v]],
  {u → map[x, y], v → domain[eval[w]]}]]
```

Out[22]=
```
or[not[member[w, x]], subclass[map[x, y], domain[eval[w]]]] == True
```

In[23]:=
```
or[not[member[w_, x_]], subclass[map[x_, y_], domain[eval[w_]]]] := True
```

In[24]:=
```
SubstTest[implies, member[w, x],
  subclass[map[x, y], domain[eval[w]]], {w → set[0], x → omega, y → Z}] // Reverse
```

Out[24]=
```
subclass[map[omega, Z], domain[eval[set[0]]]] == True
```

In[25]:=
```
subclass[map[omega, Z], domain[eval[set[0]]]] := True
```

In[26]:=
```
SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
  {u → binhom[NATADD, INTADD], v → map[omega, Z], w → domain[eval[set[0]]]}] // Reverse
```

Out[26]=
```
subclass[binhom[NATADD, INTADD], domain[eval[set[0]]]] == True
```

In[27]:=
```
subclass[binhom[NATADD, INTADD], domain[eval[set[0]]]] := True
```

In[28]:=
```
equal[intersection[binhom[NATADD, INTADD], domain[eval[set[0]]]],
 binhom[NATADD, INTADD]]
```

Out[28]=
```
True
```

In[29]:=
```
intersection[binhom[NATADD, INTADD], domain[eval[set[0]]]] := binhom[NATADD, INTADD]
```

```
In[30]:= SubstTest[domain, composite[eval[set[0]], id[x]], x → binhom[NATADD, INTADD]] // Reverse
Out[30]= range[MIXTIMES] == binhom[NATADD, INTADD]

In[31]:= range[MIXTIMES] := binhom[NATADD, INTADD]
```

sethood results

Theorem.

```
In[32]:= SubstTest[implies, and[subclass[u, v], member[v, V]],
           member[u, V], {u -> binhom[NATADD, INTADD], v -> map[omega, Z]}] // Reverse
Out[32]= member[binhom[NATADD, INTADD], V] == True

In[33]:= member[binhom[NATADD, INTADD], V] := True
```

Theorem.

```
In[34]:= member[MIXTIMES, V] // AssertTest
Out[34]= member[MIXTIMES, V] == True

In[35]:= member[MIXTIMES, V] := True
```

membership rule

Lemma.

```
In[36]:= SubstTest[member, pair[t, x], composite[Id, z], {z → MIXTIMES, w → V}]
Out[36]= and[member[t, V], member[x, V], member[pair[t, x], MIXTIMES]] ==
          member[pair[t, x], MIXTIMES]

In[37]:= and[member[t_, V], member[x_, V], member[pair[t_, x_], MIXTIMES]] :=
          member[pair[t, x], MIXTIMES]
```

Theorem. (A secondary membership rule.)

```
In[38]:= SubstTest[member, pair[t, x],
           composite[eval[set[0]], id[z]], z -> binhom[NATADD, INTADD]] // Reverse
Out[38]= member[pair[x, t], MIXTIMES] == and[equal[image[t, set[set[0]]], set[x]],
           member[t, binhom[NATADD, INTADD]], member[x, V]]

In[39]:= member[pair[x_, t_], MIXTIMES] := and[equal[image[t, set[set[0]]], set[x]],
           member[t, binhom[NATADD, INTADD]], member[x, V]]
```

lemma: each binary hom takes 0 to +0

Lemma.

```
In[40]:= SubstTest[implies, member[w, binhom[x, y]],
           subclass[image[w, fix[composite[x, DUP]]], fix[composite[y, DUP]]],
           {x → NATADD, y → INTADD}] // Reverse

Out[40]= or[not[member[w, binhom[NATADD, INTADD]]],
           subclass[image[w, set[0]], set[id[omega]]]] == True

In[41]:= (% /. w → w_) /. Equal → SetDelayed
```

Lemma.

```
In[42]:= SubstTest[implies, and[subclass[t, set[v]], not[empty[t]]],
           equal[t, set[v]], t → image[x, set[u]]] // Reverse

Out[42]= or[equal[image[x, set[u]], set[v]],
           not[member[u, domain[x]]], not[subclass[image[x, set[u]], set[v]]]] == True

In[43]:= (% /. {x → x_, u → u_, v → v_}) /. Equal → SetDelayed
```

Theorem.

```
In[44]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            implies[p1, p4], implies[and[p3, p4], p5], not[implies[p1, p5]],
            {p1 -> member[w, binhom[NATADD, INTADD]], p2 → equal[domain[w], omega],
             p3 → member[0, domain[w]], p4 -> subclass[image[w, set[0]], set[id[omega]]],
             p5 → equal[image[w, set[0]], set[id[omega]]]}]] // Reverse

Out[44]= or[equal[image[w, set[0]], set[id[omega]]],
           not[member[w, binhom[NATADD, INTADD]]]] == True

In[45]:= or[equal[image[w_, set[0]], set[id[omega]]],
           not[member[w_, binhom[NATADD, INTADD]]]] := True
```

a general result about functions

Lemma.

```
In[46]:= or[equal[y, APPLY[funpart[z], x]], not[equal[set[y], set[APPLY[funpart[z], x]]]],
           not[member[x, domain[funpart[z]]]]] // AssertTest

Out[46]= or[equal[y, APPLY[funpart[z], x]], not[equal[set[y], set[APPLY[funpart[z], x]]]],
           not[member[x, domain[funpart[z]]]]] == True

In[47]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. (Correspondence between vertical sections and application for functions. This is needed for the iteration condition in the next section.)

```
In[48]:= SubstTest[implies, equal[t, funpart[z]],
           or[equal[y, APPLY[t, x]], not[equal[image[t, set[x]], set[y]]],
            not[FUNCTION[t]], not[member[x, domain[t]]]], t → z] // Reverse

Out[48]= or[equal[y, APPLY[z, x]], not[equal[image[z, set[x]], set[y]]],
           not[FUNCTION[z]], not[member[x, domain[z]]]] == True

In[49]:= or[equal[APPLY[z_, x_], y_], not[equal[image[z_, set[x_]], set[y_]]],
           not[FUNCTION[z_]], not[member[x_, domain[z_]]]] := True
```

---

uniqueness theorem

Lemma.

```
In[50]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
           {u -> composite[t, NATADD], v -> composite[INTADD, cross[t, t]], w → RIGHT[set[0]]}] //
          Reverse

Out[50]= or[equal[composite[t, id[omega], SUCC],
            composite[INTADD, id[cart[V, image[t, set[set[0]]]], inverse[FIRST], t]],
           not[equal[composite[INTADD, cross[t, t]], composite[t, NATADD]]]] == True

In[51]:= (% /. t → t_) /. Equal → SetDelayed
```

Lemma. (Introduce **funpart** wrappers.)

```
In[52]:= SubstTest[or, equal[composite[t, id[omega], SUCC],
            composite[INTADD, id[cart[V, image[t, set[set[0]]]], inverse[FIRST], t]],
           not[equal[composite[INTADD, cross[t, t]], composite[t, NATADD]]],
           t → funpart[z]] // Reverse

Out[52]= or[equal[composite[funpart[z], id[omega], SUCC], composite[
             image[inverse[INTADD], set[APPLY[funpart[z], set[0]]]], INVERSE, funpart[z]]],
           not[equal[composite[INTADD, cross[funpart[z], funpart[z]]],
             composite[funpart[z], NATADD]]]] == True

In[53]:= (% /. z → z_) /. Equal → SetDelayed
```

Lemma. (Remove the **funpart** wrappers.)

```
In[54]:= SubstTest[implies, and[equal[t, funpart[z]], equal[x, APPLY[t, set[0]]]],
           or[equal[composite[t, id[omega], SUCC],
             composite[image[inverse[INTADD], set[x]], INVERSE, t]],
            not[equal[composite[INTADD, cross[t, t]], composite[t, NATADD]]]], z → t] // Reverse

Out[54]= or[equal[composite[t, id[omega], SUCC],
            composite[image[inverse[INTADD], set[x]], INVERSE, t]],
           not[equal[x, APPLY[t, set[0]]]],
           not[equal[composite[INTADD, cross[t, t]], composite[t, NATADD]]],
           not[FUNCTION[t]]] == True

In[55]:= (% /. t → t_) /. Equal → SetDelayed
```

Theorem. (Iteration condition)

```
In[56]:= Map[not, SubstTest[and, implies[p1, p4], implies[p1, p5],
             implies[p4, p6], implies[and[p3, p5, p7], p8], not[implies[and[p1, p2], p9]],
             {p1 -> member[t, binhom[NATADD, INTADD]], p2 → equal[image[t, set[set[0]]], set[x]],
              p3 -> equal[composite[t, NATADD], composite[INTADD, cross[t, t]]],
              p4 → equal[domain[t], omega],
              p5 → FUNCTION[t], p6 -> member[set[0], domain[t]], p7 → equal[x, APPLY[t, set[0]]],
              p8 -> equal[composite[t, id[omega], SUCC], composite[
                 image[inverse[INTADD], set[x]], INVERSE, t]], p9 -> equal[composite[t, SUCC],
                composite[image[inverse[INTADD], set[x]], INVERSE, t]]}]] // Reverse

Out[56]= or[equal[composite[t, SUCC], composite[image[inverse[INTADD], set[x]], INVERSE, t]],
           not[equal[image[t, set[set[0]]], set[x]]],
           not[member[t, binhom[NATADD, INTADD]]]] == True

In[57]:= or[equal[composite[t_, SUCC], composite[image[inverse[INTADD], set[x_]], INVERSE, t_]],
           not[equal[image[t_, set[set[0]]], set[x_]]],
           not[member[t_, binhom[NATADD, INTADD]]]] := True
```

Theorem. Application of the uniqueness theorem for **iterate**.

```
In[58]:= Map[not, SubstTest[and, implies[and[p1, p2], p4], implies[p1, p5],
             implies[and[p3, p6, p7], p8], not[implies[and[p1, p2], p8]],
             {p1 -> member[t, binhom[NATADD, INTADD]], p2 → equal[image[t, set[set[0]]], set[x]],
              p3 → equal[domain[t], omega],
              p4 ->
               equal[composite[t, SUCC], composite[image[inverse[INTADD], set[x]], INVERSE, t]],
              p5 -> equal[image[t, set[0]], set[id[omega]]],
              p6 → equal[composite[t, id[omega]],
                iterate[composite[image[inverse[INTADD], set[x]], INVERSE], set[id[omega]]]],
              p7 → FUNCTION[t],
              p8 -> equal[t, iterate[composite[image[inverse[INTADD], set[x]], INVERSE],
                 set[id[omega]]]]}]] // Reverse

Out[58]= or[equal[t,
            iterate[composite[image[inverse[INTADD], set[x]], INVERSE], set[id[omega]]]],
           not[equal[image[t, set[set[0]]], set[x]]],
           not[member[t, binhom[NATADD, INTADD]]]] == True
```

```
In[59]:= or[equal[iterate[composite[image[inverse[INTADD], set[x_]], INVERSE], set[id[omega]]],
            t_], not[equal[image[t_, set[set[0]]], set[x_]]],
          not[member[t_, binhom[NATADD, INTADD]]]] := True
```

Eliminating the variable **t** yields:

```
In[60]:= Map[equal[V, #] &,
          SubstTest[class, t, implies[member[pair[x, t], u], equal[v, t]], {u → MIXTIMES,
            v -> iterate[composite[image[inverse[INTADD], set[x]], INVERSE], set[id[omega]]]}]]

Out[60]= subclass[image[MIXTIMES, set[x]], set[
            iterate[composite[image[inverse[INTADD], set[x]], INVERSE], set[id[omega]]]]] == True

In[61]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[62]:= SubstTest[implies, subclass[u, set[v]], or[empty[u], member[u, range[SINGLETON]]],
           {u -> image[MIXTIMES, set[x]], v -> iterate[
              composite[image[inverse[INTADD], set[x]], INVERSE], set[id[omega]]]}] // Reverse

Out[62]= or[member[x, domain[funpart[MIXTIMES]]], not[member[x, Z]]] == True

In[63]:= (% /. x → x_) /. Equal → SetDelayed
```

Eliminating the variable **x**, one finds:

```
In[64]:= Map[equal[V, #] &, SubstTest[class, x,
           or[member[x, u], not[member[x, v]]], {u -> domain[funpart[MIXTIMES]], v → Z}]]

Out[64]= subclass[Z, domain[funpart[MIXTIMES]]] == True

In[65]:= % /. Equal → SetDelayed
```

Main theorem.

```
In[66]:= SubstTest[subclass, domain[w], domain[funpart[w]], w → MIXTIMES]

Out[66]= FUNCTION[MIXTIMES] == True

In[67]:= FUNCTION[MIXTIMES] := True
```

cardinality of binhom[NATADD,INTADD]

Since **MIXTIMES** is a one-to-one correspondence, its domain and range are equipollent.

```
In[68]:= SubstTest[implies, member[w, BIJ],
           member[pair[domain[w], range[w]], Q], w → MIXTIMES] // Reverse

Out[68]= member[pair[Z, binhom[NATADD, INTADD]], Q] == True
```

```
In[69]:= member[pair[Z, binhom[NATADD, INTADD]], Q] := True
```

Corollary. The set of binary homomorphisms from **NATADD** to **INTADD** is countably infinite.

```
In[70]:= SubstTest[implies,
          and[member[pair[u, v], Q], equal[omega, card[u]]], equal[omega, card[v]],
          {u → Z, v → binhom[NATADD, INTADD]}]

Out[70]= True == equal[omega, card[binhom[NATADD, INTADD]]]

In[71]:= card[binhom[NATADD, INTADD]] := omega
```

mapping properties

Theorem.

```
In[72]:= SubstTest[member, MIXTIMES, intersection[u, v, w],
           {u → FUNS, v → image[inverse[IMAGE[FIRST]], set[Z]],
            w → image[inverse[IMAGE[SECOND]], P[binhom[NATADD, INTADD]]]}] // Reverse

Out[72]= member[MIXTIMES, map[Z, binhom[NATADD, INTADD]]] == True

In[73]:= member[MIXTIMES, map[Z, binhom[NATADD, INTADD]]] := True
```

Corollary.

```
In[74]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
           {u → MIXTIMES, v → map[Z, binhom[NATADD, INTADD]], w → map[Z, map[omega, Z]]}] // Reverse

Out[74]= member[MIXTIMES, map[Z, map[omega, Z]]] == True

In[75]:= member[MIXTIMES, map[Z, map[omega, Z]]] := True
```

APPLY rules

Theorem.

```
In[76]:= SubstTest[image, funpart[w], set[x], w → MIXTIMES] // Reverse

Out[76]= image[MIXTIMES, set[x]] == set[APPLY[MIXTIMES, x]]

In[77]:= image[MIXTIMES, set[x_]] := set[APPLY[MIXTIMES, x]]
```

Theorem.

```
In[78]:= SubstTest[image, funpart[w], set[x], w → inverse[MIXTIMES]] // Reverse

Out[78]= image[inverse[MIXTIMES], set[x]] == set[APPLY[inverse[MIXTIMES], x]]
```

```
In[79]:= image[inverse[MIXTIMES], set[x_]] := set[APPLY[inverse[MIXTIMES], x]]
```