a formula for the endomorphisms of INTADD

Johan G. F. Belinfante
2006 December 23

```
In[1]:= SetDirectory["l:"]; << goedel88.21b; << tools.m

        :Package Title: goedel88.21b        2006 December 21 at midnite

        It is now:  2006 Dec 23 at 5:11

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 December 17

        weightlimit = 40
```

summary

Every endomorphism of **INTADD** is of the form **x = composite[INTMUL,LEFT[v]]** where **v** is the value of **x** at the integer **+1**.

APPLY formulas

Since **INTTIMES** is a function, its vertical sections can be expressed in terms of **APPLY**.

```
In[2]:= SubstTest[image, funpart[w], set[x], w → INTTIMES] // Reverse

Out[2]= image[INTTIMES, set[x]] == set[APPLY[INTTIMES, x]]

In[3]:= image[INTTIMES, set[x_]] := set[APPLY[INTTIMES, x]]
```

Since **INTTIMES** is one-to-one, the same is true for its inverse.

```
In[4]:= SubstTest[image, funpart[w], set[x], w → inverse[INTTIMES]] // Reverse

Out[4]= image[inverse[INTTIMES], set[x]] == set[APPLY[inverse[INTTIMES], x]]

In[5]:= image[inverse[INTTIMES], set[x_]] := set[APPLY[inverse[INTTIMES], x]]
```

The first of these formulas will later be replaced with a different one involving **INTMUL**.

fix formula

The already developed arithmetic of natural numbers can serve as a guide to derivation of formulas for integer arithmetic. The rewrite rules for **INTTIMES** will be patterned to a large extent on corresponding rules for **TIMES**. In this section, a fixed point characterization of the set **binhom[INTADD, INTADD]** is derived which is analogous to the following characterization of **binhom[NATADD, NATADD]**.

```
In[6]:= fix[composite[TIMES, eval[set[0]]]]

Out[6]= binhom[NATADD, NATADD]
```

Lemma. (Inclusion in one direction.)

```
In[7]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
          {u -> id[binhom[INTADD, INTADD]],
           v -> composite[inverse[eval[plus[set[0]]]], eval[plus[set[0]]]],
           w -> id[binhom[INTADD, INTADD]]}] // Reverse

Out[7]= subclass[binhom[INTADD, INTADD],
          fix[composite[INTTIMES, eval[composite[id[omega], SUCC]]]]] == True

In[8]:= % /. Equal → SetDelayed
```

Lemma.

```
In[9]:= SubstTest[subclass, fix[composite[x, y]],
          range[x], {x → INTTIMES, y → eval[plus[set[0]]]}] // Reverse

Out[9]= subclass[fix[composite[INTTIMES, eval[composite[id[omega], SUCC]]]],
          binhom[INTADD, INTADD]] == True

In[10]:= % /. Equal → SetDelayed
```

Theorem.

```
In[11]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> fix[composite[INTTIMES, eval[plus[set[0]]]]], v -> binhom[INTADD, INTADD]}]

Out[11]= equal[binhom[INTADD, INTADD],
           fix[composite[INTTIMES, eval[composite[id[omega], SUCC]]]]] == True

In[12]:= fix[composite[INTTIMES, eval[composite[id[omega], SUCC]]]] := binhom[INTADD, INTADD]
```

APPLY[INTTIMES, x] formula

In this section a formula for **APPLY[INTTIMES, x]** is derived which is analogous to the following formula:

```
In[13]:= APPLY[TIMES, x]

Out[13]= union[complement[image[V, intersection[omega, set[x]]]], times[x]]
```

Lemma.

```
In[14]:= SubstTest[implies, equal[w, V], empty[funpart[w]], w → APPLY[INTTIMES, x]] // Reverse

Out[14]= or[equal[0, funpart[APPLY[INTTIMES, x]]], member[x, Z]] == True

In[15]:= (% /. x → x_) /. Equal → SetDelayed
```

The following corollary of a **FUNPART** theorem is needed:

```
In[16]:= Map[equal[#, APPLY[INTTIMES, x]] &, ApComp[FUNPART, INTTIMES, x]]

Out[16]= or[FUNCTION[APPLY[INTTIMES, x]], not[member[x, Z]]] == True

In[17]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[18]:= SubstTest[implies, FUNCTION[t], not[equal[t, V]], t → APPLY[INTTIMES, x]] // Reverse

Out[18]= or[member[x, Z], not[FUNCTION[APPLY[INTTIMES, x]]]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining the preceding two results yields this temporary rewrite rule:

```
In[20]:= equiv[FUNCTION[APPLY[INTTIMES, x]], member[x, Z]]

Out[20]= True

In[21]:= FUNCTION[APPLY[INTTIMES, x_]] := member[x, Z]
```

A sethood lemma:

```
In[22]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
           {u -> funpart[APPLY[INTTIMES, x]],
            v -> intersection[image[V, intersection[Z, set[x]]], APPLY[INTTIMES, x]]}] // Reverse

Out[22]= member[funpart[APPLY[INTTIMES, x]], V] == True

In[23]:= (% /. x → x_) /. Equal → SetDelayed
```

The key step of the derivation is this reification result, which is also made into a temporary rewrite rule.

```
In[24]:= Map[reify[w, #] &, ImageComp[eval[w], INTTIMES, set[x]]] // Reverse

Out[24]= funpart[APPLY[INTTIMES, x]] == composite[INTMUL, LEFT[x]]

In[25]:= funpart[APPLY[INTTIMES, x_]] := composite[INTMUL, LEFT[x]]
```

Corollary. (Another temporary rewrite rule.)

```
In[26]:= SubstTest[equal, w, funpart[w], w -> APPLY[INTTIMES, x]] // Reverse

Out[26]= equal[APPLY[INTTIMES, x], composite[INTMUL, LEFT[x]]] == member[x, Z]

In[27]:= equal[APPLY[INTTIMES, x_], composite[INTMUL, LEFT[x_]]] := member[x, Z]
```

Main theorem. The following rewrite rule will be made permanent.

```
In[28]:= equal[APPLY[INTTIMES, x],
           union[complement[image[V, intersection[Z, set[x]]]], funpart[APPLY[INTTIMES, x]]]]

Out[28]= True

In[29]:= APPLY[INTTIMES, x_] :=
           union[complement[image[V, intersection[Z, set[x]]]], composite[INTMUL, LEFT[x]]]
```

a replacement rule

The temporary rewrite rule is now removed:

```
In[30]:= image[INTTIMES, set[x_]] =.
```

The following is a replacement for the removed rule:

```
In[31]:= SubstTest[image, funpart[w], set[x], w → INTTIMES] // Reverse

Out[31]= image[INTTIMES, set[x]] ==
           intersection[image[V, intersection[Z, set[x]]], set[composite[INTMUL, LEFT[x]]]]

In[32]:= image[INTTIMES, set[x_]] :=
           intersection[image[V, intersection[Z, set[x]]], set[composite[INTMUL, LEFT[x]]]]
```

endomorphisms of INTADD

In this section it is shown that **composite[INTMUL, LEFT[x]]** is an endomorphism of **INTADD** if and only if **x** is an integer. In one direction one has:

```
In[33]:= SubstTest[implies, member[t, binhom[INTADD, INTADD]],
             equal[domain[t], Z], t -> composite[INTMUL, LEFT[x]]] // Reverse

Out[33]= or[member[x, Z], not[member[composite[INTMUL, LEFT[x]], binhom[INTADD, INTADD]]]] == True

In[34]:= (% /. x → x_) /. Equal → SetDelayed
```

In the other direction, one has:

```
In[35]:= SubstTest[implies, and[FUNCTION[w], member[x, domain[w]]],
           member[APPLY[w, x], range[w]], w → INTTIMES] // Reverse

Out[35]= or[member[composite[INTMUL, LEFT[x]], binhom[INTADD, INTADD]], not[member[x, Z]]] == True

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining these two results yields this theorem, which is made into a rewrite rule.

```
In[37]:= equiv[member[composite[INTMUL, LEFT[x]], binhom[INTADD, INTADD]], member[x, Z]]

Out[37]= True

In[38]:= member[composite[INTMUL, LEFT[x_]], binhom[INTADD, INTADD]] := member[x, Z]
```

uniqueness theorem

Lemma.

```
In[39]:= Map[implies[#, equal[x,
              composite[INTMUL, LEFT[APPLY[funpart[x], composite[id[omega], SUCC]]]]]] &,
           (member[x, fix[composite[funpart[u], funpart[v]]]] // AssertTest)] /.
         {u → INTTIMES, v → eval[plus[set[0]]]}

Out[39]= or[equal[x, composite[INTMUL, LEFT[APPLY[funpart[x], composite[id[omega], SUCC]]]]],
           not[member[x, binhom[INTADD, INTADD]]]] == True

In[40]:= (% /. x → x_) /. Equal → SetDelayed
```

The **funpart** wrapper in the lemma is not needed because any endomorphism is a function. The result is cleaned up as follows:

```
In[41]:= (Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
             implies[and[p0, p2], p4], implies[and[p3, p4], p5], not[implies[and[p0, p1], p5]],
             {p0 → equal[x, y], p1 -> member[x, binhom[INTADD, INTADD]],
              p2 → FUNCTION[x], p3 -> equal[x, composite[INTMUL, LEFT[
                  APPLY[funpart[x], composite[id[omega], SUCC]]]]], p4 → equal[y, funpart[x]],
              p5 -> equal[x, composite[INTMUL, LEFT[APPLY[y, composite[
                    id[omega], SUCC]]]]]}]] // Reverse) /. y → x

Out[41]= or[equal[x, composite[INTMUL, LEFT[APPLY[x, composite[id[omega], SUCC]]]]],
           not[member[x, binhom[INTADD, INTADD]]]] == True

In[42]:= (% /. x → x_) /. Equal → SetDelayed
```

In the reverse direction one needs to add the possibility that **x** might be empty. One then obtains the following theorem which can be made into a rewrite rule. This result says that every endomorphism **x** of **INTADD** is of the form **composite[-INTMUL,LEFT[v]]** where **v** is the value of **x** at the integer **+1 = composite[id[omega], SUCC]**.

```
In[43]:= equiv[equal[x, composite[INTMUL, LEFT[APPLY[x, composite[id[omega], SUCC]]]]],
           or[empty[x], member[x, binhom[INTADD, INTADD]]]] // not // not

Out[43]= True

In[44]:= equal[x_, composite[INTMUL, LEFT[APPLY[x_, composite[id[omega], SUCC]]]]] :=
          or[equal[0, x], member[x, binhom[INTADD, INTADD]]]
```