INTTIMES is a function

Johan G. F. Belinfante
2006 December 21

```
In[1]:=  SetDirectory["l:"]; << goedel88.20a; << tools.m

       :Package Title: goedel88.20a       2006 December 20 at 10:45 a.m.

       It is now:  2006 Dec 21 at 11:59

       Loading Simplification Rules

       TOOLS.M                     Revised 2006 December 17

       weightlimit = 40
```

summary

It is shown in this notebook that there is a one-to-one correspondence between integers and endomorphisms of integer addition. For each integer **z** there exists a unique endomorphism of **INTADD** whose value at the integer **+1** is **z**. Using the terminology introduced the notebook **ztimes-1.nb**, this fact is equivalent to the statement that the correspondence **INTTIMES** is a function. The derivation of these facts presented here reduces the uniqueness of endomorphisms of **INTADD** to properties of mixed multiplication, for which a corresponding uniqueness theorem was earlier established (using an argument involving the uniqueness of iteration). Thus one may regard the present derivation as a somewhat indirect application of induction.

reduction to MIXMUL: positive part

The composite of binary homomorphisms is a binary homomorphism. Since **PLUS** is a homomorphism from **NATADD** to **INTADD**, composition of an endomorphism of **INTADD** with **PLUS** yields a homomorphism from **NATADD** to **INTADD**.

```
In[2]:=  SubstTest[implies, and[member[t, binhom[y, z]], member[u, binhom[x, y]]],
           member[composite[t, u], binhom[x, z]],
           {u → PLUS, x → NATADD, y → INTADD, z → INTADD}] // Reverse

Out[2]=  or[member[composite[t, PLUS], binhom[NATADD, INTADD]],
           not[member[t, binhom[INTADD, INTADD]]]] == True

In[3]:=  (% /. t → t_) /. Equal → SetDelayed
```

Homomorphisms from **NATADD** to **INTADD** are uniquely determined by their value at the natural number **1 = set[0]**. Applying this fact to the composite considered above yields:

```
In[4]:=  Map[implies[member[composite[x, PLUS], binhom[NATADD, INTADD]], #] &, SubstTest[equal, y,
            composite[MIXMUL, LEFT[APPLY[y, set[0]]]], y -> composite[x, PLUS]]] // Reverse

Out[4]=  or[equal[composite[MIXMUL, LEFT[APPLY[x, composite[id[omega], SUCC]]]],
            composite[x, PLUS]], not[member[composite[x, PLUS], binhom[NATADD, INTADD]]]] == True

In[5]:=  (% /. x → x_) /. Equal → SetDelayed
```

One can derive an explicit formula for the restriction of an endomorphism of **INTADD** to the set of non-negative integers.

```
In[6]:=  Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
            not[implies[p1, p4]], {p1 -> member[x, binhom[INTADD, INTADD]],
             p2 -> member[composite[x, PLUS], binhom[NATADD, INTADD]], p3 -> equal[composite[
                MIXMUL, LEFT[APPLY[x, composite[id[omega], SUCC]]]], composite[x, PLUS]],
             p4 -> equal[composite[MIXMUL, LEFT[APPLY[x, composite[id[omega], SUCC]]],
                inverse[PLUS]], composite[x, id[range[PLUS]]]]}]] // Reverse

Out[6]=  or[equal[composite[x, id[range[PLUS]]],
            composite[MIXMUL, LEFT[APPLY[x, composite[id[omega], SUCC]]], inverse[PLUS]]],
           not[member[x, binhom[INTADD, INTADD]]]] == True

In[7]:=  (% /. x → x_) /. Equal → SetDelayed
```

Corollary. The restriction of an endomorphism to the set of non-negative integers is uniquely determined by its value at **plus[set[0]]**.

```
In[8]:=  (Map[not, SubstTest[and, implies[and[p1, p2], p3],
              implies[and[p1, p2], p4], not[implies[and[p1, p2], p5]],
              {p1 → and[member[x, binhom[INTADD, INTADD]], member[y, binhom[INTADD, INTADD]]],
               p2 → and[equal[t, APPLY[x, composite[id[omega], SUCC]]],
                 equal[t, APPLY[y, composite[id[omega], SUCC]]]], p3 → equal[
                 composite[x, id[range[PLUS]]], composite[MIXMUL, LEFT[t], inverse[PLUS]]],
               p4 -> equal[composite[y, id[range[PLUS]]],
                 composite[MIXMUL, LEFT[t], inverse[PLUS]]],
               p5 -> equal[composite[x, id[range[PLUS]]], composite[y, id[range[PLUS]]]]}]] //
            Reverse) /. t -> APPLY[y, composite[id[omega], SUCC]]

Out[8]=  or[equal[composite[x, id[range[PLUS]]], composite[y, id[range[PLUS]]]], not[
            equal[APPLY[x, composite[id[omega], SUCC]], APPLY[y, composite[id[omega], SUCC]]]],
           not[member[x, binhom[INTADD, INTADD]]], not[member[y, binhom[INTADD, INTADD]]]] == True

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

reduction to MIXMUL: negative part

For the restriction of an endomorphism to the non-positive integers, one needs to relate the value at +1 to the value at -1. The following lemma takes care of this:

```
In[10]:= SubstTest[implies, and[member[x, binhom[INTADD, INTADD]], member[z, Z]],
           equal[APPLY[x, inverse[z]], inverse[APPLY[x, z]]], z → plus[set[0]]] // Reverse

Out[10]= or[equal[APPLY[x, composite[inverse[SUCC], id[omega]]],
            inverse[APPLY[x, composite[id[omega], SUCC]]]],
           not[member[x, binhom[INTADD, INTADD]]]] == True

In[11]:= (% /. x → x_) /. Equal → SetDelayed
```

This time one needs to consider the composition of an endomorphism with the homomorphism **composite[INVERSE,-PLUS]**.

```
In[12]:= SubstTest[implies, and[member[t, binhom[y, z]], member[u, binhom[x, y]]],
           member[composite[t, u], binhom[x, z]],
           {u → composite[INVERSE, PLUS], x → NATADD, y → INTADD, z → INTADD}] // Reverse

Out[12]= or[member[composite[t, INVERSE, PLUS], binhom[NATADD, INTADD]],
           not[member[t, binhom[INTADD, INTADD]]]] == True

In[13]:= (% /. t → t_) /. Equal → SetDelayed
```

This homomorphism is determined by its value at the natural number **1 = set[0]**.

```
In[14]:= Map[implies[member[composite[x, INVERSE, PLUS], binhom[NATADD, INTADD]], #] &,
           SubstTest[equal, y, composite[MIXMUL, LEFT[APPLY[y, set[0]]]],
            y -> composite[x, INVERSE, PLUS]]] // Reverse

Out[14]= or[equal[composite[MIXMUL, LEFT[APPLY[x, composite[inverse[SUCC], id[omega]]]]],
            composite[x, INVERSE, PLUS]],
           not[member[composite[x, INVERSE, PLUS], binhom[NATADD, INTADD]]]] == True

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

An explicit formula is obtained for the restriction to non-positive integers for any endomorphism of **INTADD**.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            implies[p1, p4], implies[and[p3, p4], p5], implies[p5, p6],
            not[implies[p1, p6]], {p1 -> member[x, binhom[INTADD, INTADD]],
             p2 -> member[composite[x, INVERSE, PLUS], binhom[NATADD, INTADD]],
             p3 -> equal[composite[MIXMUL, LEFT[APPLY[x, composite[inverse[SUCC], id[omega]]]]],
               composite[x, INVERSE, PLUS]],
             p4 → equal[APPLY[x, composite[inverse[SUCC], id[omega]]],
               inverse[APPLY[x, composite[id[omega], SUCC]]]],
             p5 -> equal[composite[MIXMUL, LEFT[inverse[APPLY[x, composite[id[omega], SUCC]]]]],
               composite[x, INVERSE, PLUS]],
             p6 -> equal[composite[MIXMUL, LEFT[inverse[APPLY[x, composite[id[omega], SUCC]]]],
                inverse[PLUS], INVERSE],
               composite[x, id[image[INVERSE, range[PLUS]]]]]}]] // Reverse

Out[16]= or[equal[composite[x, id[image[INVERSE, range[PLUS]]]], composite[MIXMUL,
             LEFT[inverse[APPLY[x, composite[id[omega], SUCC]]]], inverse[PLUS], INVERSE]],
           not[member[x, binhom[INTADD, INTADD]]]] == True
```

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

If two endomorphisms of **INTADD** have the same value at the integer **1 = plus[set[0]]**, then their restrictions to non-positive integers are equal.

```
In[18]:=  (Map[not, SubstTest[and, implies[and[p1, p2], p3],
              implies[and[p1, p2], p4], not[implies[and[p1, p2], p5]],
              {p1 → and[member[x, binhom[INTADD, INTADD]], member[y, binhom[INTADD, INTADD]]],
               p2 → and[equal[t, APPLY[x, composite[id[omega], SUCC]]],
                 equal[t, APPLY[y, composite[id[omega], SUCC]]]],
               p3 → equal[composite[x, id[image[INVERSE, range[PLUS]]]],
                 composite[MIXMUL, LEFT[inverse[t]], inverse[PLUS], INVERSE]],
               p4 -> equal[composite[y, id[image[INVERSE, range[PLUS]]]],
                 composite[MIXMUL, LEFT[inverse[t]], inverse[PLUS], INVERSE]],
               p5 -> equal[composite[x, id[image[INVERSE, range[PLUS]]]],
                 composite[y, id[image[INVERSE, range[PLUS]]]]]}]] //
            Reverse) /. t -> APPLY[y, composite[id[omega], SUCC]]

Out[18]=  or[equal[composite[x, id[image[INVERSE, range[PLUS]]]],
             composite[y, id[image[INVERSE, range[PLUS]]]]], not[
             equal[APPLY[x, composite[id[omega], SUCC]], APPLY[y, composite[id[omega], SUCC]]]],
            not[member[x, binhom[INTADD, INTADD]]],
            not[member[y, binhom[INTADD, INTADD]]]] == True

In[19]:=  (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

combining the positive and negative restrictions

Lemma.

```
In[20]:=  SubstTest[implies, and[member[x, y], subclass[y, z]], member[x, z],
            {y -> binhom[INTADD, INTADD], z → P[cart[Z, V]]}] // Reverse

Out[20]=  or[not[member[x, binhom[INTADD, INTADD]]], subclass[x, cart[Z, V]]] == True

In[21]:=  (% /. x → x_) /. Equal → SetDelayed
```

If **x** and **y** have equal restrictions for both non-positive and non-negative integers, then they have equal restrictions to all integers.

```
In[22]:=  SubstTest[implies, and[equal[s, t], equal[u, v]], equal[union[s, u], union[t, v]],
            {s -> composite[x, id[range[PLUS]]], t -> composite[y, id[range[PLUS]]],
             u -> composite[x, id[image[INVERSE, range[PLUS]]]],
             v -> composite[y, id[image[INVERSE, range[PLUS]]]]}] // Reverse

Out[22]=  or[equal[composite[x, id[Z]], composite[y, id[Z]]],
            not[equal[composite[x, id[image[INVERSE, range[PLUS]]]],
              composite[y, id[image[INVERSE, range[PLUS]]]]]],
            not[equal[composite[x, id[range[PLUS]]], composite[y, id[range[PLUS]]]]]] == True
```

```
In[23]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

If endomorphisms **x** and **y** of **INTADD** have equal restrictions to all integers, then they are equal. (Comment. To prevent unwanted rewrite rules from being applied, two temporary variables **s** and **t** are introduced, and then eliminated.)

```
In[24]:= (Map[not, SubstTest[and, implies[and[p0, p1], p5],
              implies[and[p0, p1], p6], implies[and[p4, p5, p6], p7],
              not[implies[and[p0, p1, p4], p7]], {p0 → and[equal[x, s], equal[y, t]],
               p1 → and[member[x, binhom[INTADD, INTADD]], member[y, binhom[INTADD, INTADD]]],
               p4 → equal[composite[x, id[Z]], composite[y, id[Z]]],
               p5 → equal[s, composite[x, id[Z]]], p6 -> equal[t, composite[y, id[Z]]],
               p7 → equal[s, t]}]] // Reverse) /. {s → x, t → y}

Out[24]= or[equal[x, y], not[equal[composite[x, id[Z]], composite[y, id[Z]]]],
           not[member[x, binhom[INTADD, INTADD]]],
           not[member[y, binhom[INTADD, INTADD]]]] == True

In[25]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Uniqueness theorem. If two endomorphisms of **INTADD** have the same value at the integer **+1**, then they are equal.

```
In[26]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
            implies[and[p2, p3], p4], implies[and[p1, p4], p7], not[implies[p1, p7]], {p1 →
              and[member[x, binhom[INTADD, INTADD]], member[y, binhom[INTADD, INTADD]], equal[
                APPLY[x, composite[id[omega], SUCC]], APPLY[y, composite[id[omega], SUCC]]]],
             p2 → equal[composite[x, id[range[PLUS]]], composite[y, id[range[PLUS]]]],
             p3 -> equal[composite[x, id[image[INVERSE, range[PLUS]]]],
               composite[y, id[image[INVERSE, range[PLUS]]]]],
             p4 → equal[composite[x, id[Z]], composite[y, id[Z]]], p7 → equal[x, y]}]] // Reverse

Out[26]= or[equal[x, y], not[
            equal[APPLY[x, composite[id[omega], SUCC]], APPLY[y, composite[id[omega], SUCC]]]],
           not[member[x, binhom[INTADD, INTADD]]],
           not[member[y, binhom[INTADD, INTADD]]]] == True

In[27]:= or[equal[x_, y_], not[equal[APPLY[x_, composite[id[omega], SUCC]],
             APPLY[y_, composite[id[omega], SUCC]]]], not[member[x_, binhom[INTADD, INTADD]]],
           not[member[y_, binhom[INTADD, INTADD]]]] := True
```

In the next section, a variable-free version of this uniqueness theorem is derived.

eliminating variables

Observation. The following result holds with **funpart** wrappers.

```
In[28]:= implies[and[member[pair[w, funpart[x]], INTTIMES],
           member[pair[w, funpart[y]], INTTIMES]], equal[funpart[x], funpart[y]]]

Out[28]= True
```

The usual technique for removing **funpart** wrappers does not work here, presumably because of limitations on equality substitution. A different technique will be used, namely, the variables will be removed first, converting **funpart** to **FUNPART**, and then **FUNPART** will be eliminated later. To carry out this strategy, the following temporary lemma is needed.

```
In[29]:= (member[pair[x, y], composite[inverse[FUNPART], z]] // AssertTest) /. z → INTTIMES

Out[29]= member[pair[x, y], composite[inverse[FUNPART], INTTIMES]] ==
          and[equal[x, APPLY[funpart[y], composite[id[omega], SUCC]]],
           member[x, V], member[y, V], member[funpart[y], binhom[INTADD, INTADD]]]

In[30]:= member[pair[x_, y_], composite[inverse[FUNPART], INTTIMES]] :=
          and[equal[x, APPLY[funpart[y], composite[id[omega], SUCC]]],
           member[x, V], member[y, V], member[funpart[y], binhom[INTADD, INTADD]]]
```

All three variables are eliminated simultaneously as follows.

```
In[31]:= Map[empty[composite[complement[#], id[cart[V, V]]]] &,
          SubstTest[class, pair[pair[x, y], w],
           implies[and[member[pair[w, x], z], member[pair[w, y], z]], member[pair[x, y], t]],
           {z → composite[inverse[FUNPART], INTTIMES],
            t -> composite[inverse[FUNPART], FUNPART]}]]

Out[31]= subclass[composite[inverse[FUNPART], INTTIMES, inverse[INTTIMES], FUNPART],
           composite[inverse[FUNPART], FUNPART]] == True

In[32]:= % /. Equal → SetDelayed
```

Lemma.

```
In[33]:= Assoc[id[FUNS], FUNPART, INTTIMES]

Out[33]= composite[id[FUNS], INTTIMES] == INTTIMES

In[34]:= composite[id[FUNS], INTTIMES] := INTTIMES
```

Lemma.

```
In[35]:= composite[inverse[INTTIMES], id[FUNS]] // DoubleInverse

Out[35]= composite[inverse[INTTIMES], id[FUNS]] == inverse[INTTIMES]

In[36]:= composite[inverse[INTTIMES], id[FUNS]] := inverse[INTTIMES]
```

Theorem. Variable-free formulation of the uniqueness theorem for endomorphisms of **INTADD**.

```
In[37]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u -> composite[inverse[FUNPART], INTTIMES, inverse[INTTIMES], FUNPART],
            v -> composite[inverse[FUNPART], FUNPART], w → cross[FUNPART, FUNPART]}] // Reverse

Out[37]= FUNCTION[INTTIMES] == True

In[38]:= FUNCTION[INTTIMES] := True
```

mapping properties of INTTIMES

The function **INTTIMES** maps integers to endomorphisms of **INTADD**.

```
In[39]:=  member[INTTIMES, map[Z, binhom[INTADD, INTADD]]] // AssertTest

Out[39]=  member[INTTIMES, map[Z, binhom[INTADD, INTADD]]] == True

In[40]:=  member[INTTIMES, map[Z, binhom[INTADD, INTADD]]] := True
```

Corollary.

```
In[41]:=  member[INTTIMES, map[Z, map[Z, Z]]] // AssertTest

Out[41]=  member[INTTIMES, map[Z, map[Z, Z]]] == True

In[42]:=  member[INTTIMES, map[Z, map[Z, Z]]] := True
```

an alternative approach to the uniqueness theorem

The uniqueness of the zero endomorphism of **INTADD** had been derived previously in the notebook **binhom-7.nb** by a different method. Another strategy for establishing the general uniqueness theorem would be to attempt to reduce it to this special case. To do so, one would need to consider the difference between two endomorphisms **x** and **y** having the same value at **+1**. This difference is an endomorphism which has the value zero at **+1**, and therefore is equal to the zero endomorphism by the uniqueness of zero endomorphisms. One would then be faced with having to show that if the difference between two endomorphisms is the zero endomorphism, then they are equal. This alternative approach would require a somewhat more extensive development of the theory of adding and subtracting endomorphisms than has presently been achieved.