Mixed multiplication, part 2. Distributive laws.

Johan G. F. Belinfante
2006 December 14

```
In[1]:= SetDirectory["l:"]; << goedel88.13a; << tools.m

:Package Title: goedel88.13a        2006 December 13 at 11:45 p.m.

It is now:  2006 Dec 14 at 14:15

Loading Simplification Rules

TOOLS.M                      Revised 2006 December 5

weightlimit = 40
```

introduction and summary

This notebook is concerned with distributive laws for mixed multiplication **MIXMUL** of integers times natural numbers. The function **MIXMUL** satisfies two distinct distributive laws, one involving sums of integers and the other involving sums of natural numbers. Since mixed multiplication was defined in terms of binary homomorpisms from **NATADD** to **INTADD**, the latter distributive law is an immediate consequence:

```
In[2]:= composite[INTADD, cross[MIXMUL, MIXMUL], TWIST, cross[DUP, Id]]

Out[2]= composite[MIXMUL, cross[Id, NATADD]]
```

In this notebook, the other distributive law is derived as a consequence of the fact that sums of homomorphisms from **NATADD** to **INTADD** are homomorphisms and that any such homomorphism is uniquely determined by its value at the natural number **1 = set[0]**.

map[omega, Z] is closed under sums

In this section it is shown that the sum of any two mappings from **omega** to **Z** is another one. If **f** and **g** are functions from **omega** to **Z**, then their **sum** is the function **h = f + g** defined by **h(n) = f(n) + g(n)**. Note that the natural number variable **n** occurs only once on the left, but twice on the right. The **sum** of mappings **f** and **g** is defined to be **composite[INTADD, cross[f, g], DUP]**. The following termporary abbreviation will be used:

```
In[3]:= intsum[x_, y_] := composite[INTADD,
          intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]
```

Lemma. The sum is a set.

```
In[4]:= or[member[composite[INTADD,
            intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]], V],
          not[member[x, u]], not[member[y, v]]] // AssertTest

Out[4]= or[member[composite[INTADD,
            intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]], V],
          not[member[x, u]], not[member[y, v]]] == True

In[5]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. The sum is a function.

```
In[6]:= SubstTest[implies, and[equal[x, funpart[u]], equal[y, funpart[v]]],
          FUNCTION[composite[INTADD, intersection[composite[inverse[FIRST], x],
             composite[inverse[SECOND], y]]]], {u → x, v → y}] // Reverse

Out[6]= or[FUNCTION[composite[INTADD,
            intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]],
          not[FUNCTION[x]], not[FUNCTION[y]]] == True

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

Corollary.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[p1, p4]], {p1 → and[member[x, map[omega, Z]], member[y, map[omega, Z]]],
            p2 → FUNCTION[x], p3 → FUNCTION[y], p4 → FUNCTION[composite[INTADD, intersection[
                composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]]}]] // Reverse

Out[8]= or[FUNCTION[composite[INTADD,
            intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]],
          not[member[x, map[omega, Z]]], not[member[y, map[omega, Z]]]] == True

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

Lemma.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 -> member[x, map[omega, Z]], p2 -> equal[domain[x], omega], p3 ->
              subclass[range[x], Z], p4 -> equal[domain[x], image[inverse[x], Z]]}]] // Reverse

Out[10]= or[equal[domain[x], image[inverse[x], Z]], not[member[x, map[omega, Z]]]] == True

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

Lemma about the domain. Comment: this derivation is expedited by omitting the following steps of the proof: **implies[and[p2,p5],p7], implies[and[p1,p4],p6]**, **implies[p1,p4]**, **implies[p2,p5]**. Adding these steps increases the execution time from 5.9 seconds to 8.7 seconds.

```
In[12]:= Map[not, SubstTest[and, implies[and[p3, p4, p5, p6, p7], p8],
            not[implies[and[p1, p2, p3], p8]], {p1 → member[x, map[omega, Z]],
             p2 → member[y, map[omega, Z]], p3 → equal[z, composite[INTADD,
                intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]],
             p4 → equal[domain[x], omega], p5 → equal[domain[y], omega],
             p6 → equal[domain[x], image[inverse[x], Z]],
             p7 → equal[domain[y], image[inverse[y], Z]], p8 → equal[domain[z], omega]}]] /.
          z -> composite[INTADD, intersection[composite[inverse[FIRST], x],
             composite[inverse[SECOND], y]]] // Reverse

Out[12]= or[equal[omega, intersection[image[inverse[x], Z], image[inverse[y], Z]]],
          not[member[x, map[omega, Z]]], not[member[y, map[omega, Z]]]] == True

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

Theorem. The set **map[omega, Z]** is closed under the sum operation:

```
In[14]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
            implies[and[p1, p2], p4], implies[and[p1, p2], p5], implies[and[p1, p2], p6],
            implies[and[p3, p4, p5, p6], p7], not[implies[and[p1, p2], p7]],
            {p1 → and[member[x, map[omega, Z]], member[y, map[omega, Z]]],
             p2 -> equal[z, composite[INTADD,
                intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]],
             p3 → member[z, V], p4 → FUNCTION[z], p5 → equal[omega, domain[z]],
             p6 → subclass[range[z], Z], p7 → member[z, map[omega, Z]]}]] /.
          z -> composite[INTADD, intersection[composite[inverse[FIRST], x],
             composite[inverse[SECOND], y]]] // Reverse

Out[14]= or[member[composite[INTADD, intersection[
             composite[inverse[FIRST], x], composite[inverse[SECOND], y]]], map[omega, Z]],
          not[member[x, map[omega, Z]]], not[member[y, map[omega, Z]]]] == True

In[15]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

sums of homomorphisms

A homomorphism **t** from **NATADD** to **INTADD** is a mapping from **omega** to **Z** that satisfies **composite[t, NATADD] = composite[INTADD, cross[t,t]**. In this section it is shown that the sum of two homomorphisms is another. The commutative and associative laws for **INTADD** imply the following twist rule which forms the basis of the derivation in this section.

```
In[16]:= composite[INTADD, cross[INTADD, INTADD], TWIST]

Out[16]= composite[INTADD, cross[INTADD, INTADD]]
```

Lemma.

```
In[17]:= Map[
           equal[composite[INTADD, intersection[composite[inverse[FIRST], INTADD, cross[x, x]],
               composite[inverse[SECOND], INTADD, cross[y, y]]]], composite[#, DUP]] &,
           Assoc[composite[INTADD, cross[INTADD, INTADD]], composite[TWIST,
             cross[cross[x, y], cross[x, y]]], TWIST]] // Reverse

Out[17]= equal[
           composite[INTADD, cross[composite[INTADD, intersection[composite[inverse[FIRST], x],
               composite[inverse[SECOND], y]]], composite[INTADD,
              intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]]],
           composite[INTADD, intersection[composite[inverse[FIRST], INTADD, cross[x, x]],
             composite[inverse[SECOND], INTADD, cross[y, y]]]]] == True

In[18]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[19]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4],
             implies[p4, p5], not[implies[and[p1, p2, p3], p5]],
             {p1 -> equal[composite[x, NATADD], composite[INTADD, cross[x, x]]],
              p2 -> equal[composite[y, NATADD], composite[INTADD, cross[y, y]]],
              p3 → equal[w, composite[INTADD, cross[x, y], DUP, NATADD]],
              p4 → equal[w, composite[INTADD,
                 cross[INTADD, INTADD], cross[cross[x, x], cross[y, y]], DUP]],
              p5 -> equal[w, composite[INTADD, cross[INTADD, INTADD],
                 cross[cross[x, y], cross[x, y]], cross[DUP, DUP]]]}]] /.
           w -> composite[INTADD, cross[x, y], DUP, NATADD] // Reverse

Out[19]= or[equal[composite[INTADD,
             cross[composite[INTADD, intersection[composite[inverse[FIRST], x],
                composite[inverse[SECOND], y]]], composite[INTADD,
               intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]]]],
            composite[INTADD, intersection[composite[inverse[FIRST], x],
              composite[inverse[SECOND], y]], NATADD]],
           not[equal[composite[INTADD, cross[x, x]], composite[x, NATADD]]],
           not[equal[composite[INTADD, cross[y, y]], composite[y, NATADD]]]] == True

In[20]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[21]:= SubstTest[implies, and[member[w, map[fix[domain[x]], fix[domain[y]]]],
            equal[composite[w, x], composite[y, cross[w, w]]]],
           member[w, binhom[x, y]], {x → NATADD, y → INTADD}] // Reverse

Out[21]= or[member[w, binhom[NATADD, INTADD]],
           not[equal[composite[INTADD, cross[w, w]], composite[w, NATADD]]],
           not[member[w, map[omega, Z]]]] == True

In[22]:= (% /. w → w_) /. Equal → SetDelayed
```

Theorem. The sum of homomorphisms is an homomorphism. (Remark: The execution time for this derivation was reduced from 93.5 seconds to 7.1 seconds by omitting the following step of the proof: **implies[and[p4, p5], p7]**. This dramatic reduction in time is presumably due to the fact that **p4** and **p5** are both equations, so omitting that step reduces the number of equality substitutions that can be attempted.)

```
In[23]:= Map[not,
           SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p1, p4], implies[p1, p5],
            implies[and[p2, p3], p6], implies[and[p6, p7], p8], not[implies[p1, p8]],
            {p1 → and[member[x, binhom[NATADD, INTADD]], member[y, binhom[NATADD, INTADD]]],
             p2 → member[x, map[omega, Z]], p3 → member[y, map[omega, Z]],
             p4 -> equal[composite[x, NATADD], composite[INTADD, cross[x, x]]],
             p5 -> equal[composite[y, NATADD], composite[INTADD, cross[y, y]]],
             p6 → member[intsum[x, y], map[omega, Z]],
             p7 -> equal[composite[intsum[x, y], NATADD],
               composite[INTADD, cross[intsum[x, y], intsum[x, y]]]],
             p8 → member[intsum[x, y], binhom[NATADD, INTADD]]}]] // Reverse
```

```
Out[23]= or[member[composite[INTADD,
              intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], y]]],
            binhom[NATADD, INTADD]], not[member[x, binhom[NATADD, INTADD]]],
           not[member[y, binhom[NATADD, INTADD]]]] == True
```

```
In[24]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

---

derivation of a distributive law

All binary homomorphisms from **NATADD** to **INTADD** are of the form **composite[MIXMUL,LEFT[x]]**. Therefore one can reformulate the theorem of the last section as follows:

```
In[25]:= SubstTest[implies, and[member[u, binhom[NATADD, INTADD]],
            member[v, binhom[NATADD, INTADD]]], member[intsum[u, v], binhom[NATADD, INTADD]],
           {u -> composite[MIXMUL, LEFT[x]], v -> composite[MIXMUL, LEFT[y]]}] // Reverse
```

```
Out[25]= or[member[composite[INTADD, intersection[composite[inverse[FIRST], MIXMUL, LEFT[x]],
              composite[inverse[SECOND], MIXMUL, LEFT[y]]]],
            binhom[NATADD, INTADD]], not[member[x, Z]], not[member[y, Z]]] == True
```

```
In[26]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. (A distributive law formulated using variables for the two integers. The uniqueness theorem for homomorphisms is used in this derivation.)

```
In[27]:= SubstTest[equal, t, composite[MIXMUL, LEFT[APPLY[t, set[0]]]],
          t → intsum[composite[MIXMUL, LEFT[x]], composite[MIXMUL, LEFT[y]]]]
```

```
Out[27]= True == equal[composite[INTADD, intersection[composite[inverse[FIRST], MIXMUL, LEFT[x]],
             composite[inverse[SECOND], MIXMUL, LEFT[y]]]],
           composite[MIXMUL, LEFT[intadd[x, y]]]]
```

```
In[28]:= composite[INTADD, intersection[composite[inverse[FIRST], MIXMUL, LEFT[x_]],
            composite[inverse[SECOND], MIXMUL, LEFT[y_]]]] :=
          composite[MIXMUL, LEFT[intadd[x, y]]]
```

Using **reify**, one can quickly eliminate the two integer variables, yielding this variable-free distributive law:

```
In[29]:= Map[rotate[inverse[#]] &, SubstTest[reify, x,
           composite[z, intersection[composite[inverse[FIRST], MIXMUL, LEFT[first[x]]],
             composite[inverse[SECOND], MIXMUL, LEFT[second[x]]]]]], z → INTADD]]

Out[29]= composite[INTADD, cross[MIXMUL, MIXMUL], TWIST, cross[Id, DUP]] ==
          composite[MIXMUL, cross[INTADD, Id]]

In[30]:= composite[INTADD, cross[MIXMUL, MIXMUL], TWIST, cross[Id, DUP]] :=
          composite[MIXMUL, cross[INTADD, Id]]
```

Note that the right side of this rewrite rule involves **INTADD**. The other distributive law mentioned in the introduction had **NATADD** on the right hand side. Both distributive laws have **INTADD** on the left hand side. The other difference between the two distributive laws is the position of **DUP**. For one law a natural number argument is duplicated, and for the other one, an integer argument is duplicated.