INTMUL, part 3. associative law

Johan G. F. Belinfante
2006 December 29

```
In[1]:= SetDirectory["l:"]; << goedel88.26a; << tools.m

        :Package Title: goedel88.26a        2006 December 26 at 3:50 p.m.

        It is now:  2006 Dec 29 at 22:33

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 December 17

        weightlimit = 40
```

summary

In this notebook, the associative law for integer multiplication is derived.

composites of endomorphisms

Lemma. The composite of two endomorphisms of **INTADD** is another.

```
In[2]:= SubstTest[implies,
          and[member[u, binhom[INTADD, INTADD]], member[v, binhom[INTADD, INTADD]]],
          member[composite[u, v], binhom[INTADD, INTADD]],
          {u → composite[INTMUL, LEFT[x]], v → composite[INTMUL, LEFT[y]]}] // Reverse

Out[2]= or[member[composite[INTMUL, LEFT[x], INTMUL, LEFT[y]], binhom[INTADD, INTADD]],
          not[member[x, Z]], not[member[y, Z]]] == True

In[3]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Conversely:

```
In[4]:= SubstTest[implies, member[z, binhom[INTADD, INTADD]], equal[domain[z], Z],
          z -> composite[INTMUL, LEFT[x], INTMUL, LEFT[y]]] // Reverse

Out[4]= or[and[member[x, Z], member[y, Z]], not[
           member[composite[INTMUL, LEFT[x], INTMUL, LEFT[y]], binhom[INTADD, INTADD]]]] == True

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

These two results can be combined into a single (temporary) rewrite rule.

```
In[6]:= equiv[member[composite[INTMUL, LEFT[x], INTMUL, LEFT[y]], binhom[INTADD, INTADD]],
         and[member[x, Z], member[y, Z]]]

Out[6]= True

In[7]:= member[composite[INTMUL, LEFT[x_], INTMUL, LEFT[y_]], binhom[INTADD, INTADD]] :=
         and[member[x, Z], member[y, Z]]
```

simplification rules for images of INTMUL

Lemma.

```
In[8]:= ImageComp[id[Z], INTMUL, x] // Reverse

Out[8]= intersection[Z, image[INTMUL, x]] == image[INTMUL, x]

In[9]:= intersection[Z, image[INTMUL, x_]] := image[INTMUL, x]
```

Lemma.

```
In[10]:= Assoc[INTMUL, id[cart[Z, Z]], id[cart[V, Z]]] // Reverse

Out[10]= composite[INTMUL, id[cart[V, Z]]] == INTMUL

In[11]:= composite[INTMUL, id[cart[V, Z]]] := INTMUL
```

Lemma.

```
In[12]:= ImageComp[INTMUL, id[cart[V, Z]], cart[x, y]] // Reverse

Out[12]= image[INTMUL, cart[x, intersection[y, Z]]] == image[INTMUL, cart[x, y]]

In[13]:= image[INTMUL, cart[x_, intersection[y_, Z]]] := image[INTMUL, cart[x, y]]
```

the associative law

The uniqueness theorem for endomorphisms of **INTADD** is applied to the composite of two of them.

```
In[14]:= SubstTest[equal, t, composite[INTMUL, LEFT[APPLY[t, plus[set[0]]]]],
          t -> composite[INTMUL, LEFT[x], INTMUL, LEFT[y]]]

Out[14]= True == equal[composite[INTMUL, LEFT[APPLY[INTMUL, PAIR[x, y]]]],
           composite[INTMUL, LEFT[x], INTMUL, LEFT[y]]]

In[15]:= composite[INTMUL, LEFT[x_], INTMUL, LEFT[y_]] :=
          composite[INTMUL, LEFT[APPLY[INTMUL, PAIR[x, y]]]]
```

Two applications of **reify** are used to eliminate the variables.

```
In[16]:= Map[rotate[inverse[#]] &, Map[reify[x, #] &, Map[rotate[inverse[#]] &,
           SubstTest[reify, y, composite[z, LEFT[x], INTMUL, LEFT[y]], z → INTMUL]]] // Reverse

Out[16]= composite[INTMUL, cross[INTMUL, Id], inverse[ASSOC]] ==
          composite[INTMUL, cross[Id, INTMUL]]

In[17]:= composite[INTMUL, cross[INTMUL, Id], inverse[ASSOC]] :=
          composite[INTMUL, cross[Id, INTMUL]]
```

Theorem.

```
In[18]:= Assoc[composite[INTMUL, cross[INTMUL, Id]], inverse[ASSOC], ASSOC] // Reverse

Out[18]= composite[INTMUL, cross[Id, INTMUL], ASSOC] == composite[INTMUL, cross[INTMUL, Id]]

In[19]:= composite[INTMUL, cross[Id, INTMUL], ASSOC] := composite[INTMUL, cross[INTMUL, Id]]
```

Corollary.

```
In[20]:= associative[INTMUL] // AssertTest

Out[20]= associative[INTMUL] == True

In[21]:= associative[INTMUL] := True
```

Lemma.

```
In[22]:= SubstTest[member, x, map[cartsq[fix[domain[x]]], fix[domain[x]]], x → INTMUL]

Out[22]= member[INTMUL, BINOPS] == True

In[23]:= member[INTMUL, BINOPS] := True
```

Theorem.

```
In[24]:= member[INTMUL, SEMIGPS] // AssertTest

Out[24]= member[INTMUL, SEMIGPS] == True

In[25]:= member[INTMUL, SEMIGPS] := True
```

---

rules of signs

Theorem. Multiplying an integer by minus one amounts to changing its sign.

```
In[26]:= SubstTest[equal, t, composite[INTMUL, LEFT[APPLY[t, plus[set[0]]]]],
          t -> composite[id[Z], INVERSE]]

Out[26]= True == equal[composite[INTMUL, LEFT[composite[inverse[SUCC], id[omega]]]],
           composite[id[Z], INVERSE]]
```

```
In[28]:= composite[INTMUL, LEFT[composite[inverse[SUCC], id[omega]]]] :=
          composite[id[Z], INVERSE]
```

Lemma. A simplification rule.

```
In[30]:= Assoc[INTMUL, id[cart[V, Z]], cross[x, y]]

Out[30]= composite[INTMUL, cross[x, composite[id[Z], y]]] == composite[INTMUL, cross[x, y]]

In[31]:= composite[INTMUL, cross[x_, composite[id[Z], y_]]] := composite[INTMUL, cross[x, y]]
```

Theorem. Rule of signs.

```
In[32]:= Assoc[composite[INTMUL, cross[Id, INTMUL]], ASSOC, RIGHT[inverse[plus[set[0]]]]]

Out[32]= composite[INTMUL, cross[Id, INVERSE]] == composite[INVERSE, INTMUL]

In[33]:= composite[INTMUL, cross[Id, INVERSE]] := composite[INVERSE, INTMUL]
```

Corollary.

```
In[34]:= Assoc[INTMUL, cross[Id, INVERSE], SWAP]

Out[34]= composite[INTMUL, cross[INVERSE, Id]] == composite[INVERSE, INTMUL]

In[35]:= composite[INTMUL, cross[INVERSE, Id]] := composite[INVERSE, INTMUL]
```

Lemma. A simplification rule.

```
In[37]:= Assoc[id[P[cart[V, V]]], id[Z], INTMUL]

Out[37]= composite[id[P[cart[V, V]]], INTMUL] == INTMUL

In[38]:= composite[id[P[cart[V, V]]], INTMUL] := INTMUL
```

Corollary.

```
In[39]:= Assoc[INTMUL, cross[INVERSE, Id], cross[Id, INVERSE]]

Out[39]= composite[INTMUL, cross[INVERSE, INVERSE]] == INTMUL

In[40]:= composite[INTMUL, cross[INVERSE, INVERSE]] := INTMUL
```

Comment. The orientation of all these rewrite rules should be regarded as tentative. The motivation is to bring all minus signs out in front.