INTMUL, part 1. Basics.

Johan G. F. Belinfante
2006 December 21

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

```
:Package Title: goedel88.21a        2006 December 21 at 12:15 noon

It is now:  2006 Dec 21 at 13:32

Loading Simplification Rules

TOOLS.M                      Revised 2006 December 17

weightlimit = 40
```

summary

In this notebook, the binary operation **INTMUL** for multiplication of integers is introduced, and some of its basic properties are derived. Integer multiplication is here defined by its properties, rather than by a particular construction. Specifically, an integer **z** is defined to be the product of an integer **x** and an integer **y** if there exists an addition-preserving operation **w** on the set **Z** of integers which takes the integer **+1 = plus[set[0]]** to the integer **x** and takes the integer **y** to the integer **z**. One might paraphrase this definition of multiplication loosely as follows: **x y = z** ⇔ **x : 1 = z : y**. A **class**-wrapped membership rule for this function has been introduced:

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

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

Out[3]=
```
class[u_, member[v_, INTMUL]] := ReleaseHold[Module[{w = Unique[], x = Unique[], y =
  Unique[], z = Unique[]}, class[u, exists[w, x, y, z, and[member[w, binhom[
  INTADD, INTADD]]], equal[v, pair[pair[x, y], z]], equal[set[x], image[w,
  set[composite[id[omega], SUCC]]]], equal[set[z], image[w, set[y]]]]]]]]
```

This approach to integer multiplication is patterned closely on the previously developed theory of mixed multiplication of integers by natural numbers. The strategy is to construct the theory of the binary operation **INTMUL** using previously derived facts about the function **INTTIMES**. These two functions are related to each other via currying and uncurrying. The definition of the binary operation **INTMUL** is quite similar to the definition of the curried multiplication function **INTTIMES**, but that of **INTTIMES** involves fewer variable than that of **INTMUL**. This is why **INTTIMES** has been introduced before **INTMUL**.

normalization for INTMUL

The function **INTMUL** can be related to **INTTIMES** as follows:

```
In[4]:= INTMUL // Normality // Reverse

Out[4]= composite[inverse[SINGLETON], IMG, cross[INTTIMES, SINGLETON]] == INTMUL

In[5]:= composite[inverse[SINGLETON], IMG, cross[INTTIMES, SINGLETON]] := INTMUL
```

Corollary.

```
In[6]:= Assoc[Id, composite[inverse[SINGLETON], IMG], cross[INTTIMES, SINGLETON]]

Out[6]= composite[Id, INTMUL] == INTMUL

In[7]:= composite[Id, INTMUL] := INTMUL
```

Corollary.

```
In[8]:= Assoc[composite[inverse[SINGLETON], IMG],
          cross[INTTIMES, SINGLETON], id[cart[V, V]]] // Reverse

Out[8]= composite[INTMUL, id[cart[V, V]]] == INTMUL

In[9]:= composite[INTMUL, id[cart[V, V]]] := INTMUL
```

rotation theorems

Theorem. (A variant of the formula connecting **INTMUL** with **INTTIMES**.)

```
In[10]:= Assoc[rotate[E], cross[FUNPART, Id], cross[INTTIMES, Id]]

Out[10]= composite[rotate[E], cross[INTTIMES, Id]] == INTMUL

In[11]:= composite[rotate[E], cross[INTTIMES, Id]] := INTMUL
```

Theorem. (Yet another variant.)

```
In[12]:= (composite[rotate[E], cross[x, Id]] // TripleRotate // Reverse) /. x → INTTIMES

Out[12]= rotate[composite[inverse[INTTIMES], E]] == INTMUL

In[13]:= rotate[composite[inverse[INTTIMES], E]] := INTMUL
```

domain of INTMUL

Theorem.

```
In[14]:= IminComp[composite[inverse[SINGLETON], IMG], cross[INTTIMES, SINGLETON], V]

Out[14]= domain[INTMUL] == cart[Z, Z]

In[15]:= domain[INTMUL] := cart[Z, Z]
```

INTMUL is a function

The fact that **INTTIMES** is a function implies that **INTMUL** is also a function.

```
In[16]:= SubstTest[FUNCTION, composite[funpart[x], funpart[y]],
           {x -> composite[inverse[SINGLETON], IMG], y -> cross[INTTIMES, SINGLETON]}] // Reverse

Out[16]= FUNCTION[INTMUL] == True

In[17]:= FUNCTION[INTMUL] := True
```

Corollary. (A function is a set if and only if its domain is a set.)

```
In[18]:= member[INTMUL, V] // AssertTest

Out[18]= member[INTMUL, V] == True

In[19]:= member[INTMUL, V] := True
```

range formula

```
In[20]:= ImageComp[rotate[E], cross[INTTIMES, Id], V]

Out[20]= range[INTMUL] == Z

In[21]:= range[INTMUL] := Z
```

Corollary. (Mapping property of **INTMUL**.)

```
In[22]:= member[INTMUL, map[cart[Z, Z], Z]] // AssertTest

Out[22]= member[INTMUL, map[cart[Z, Z], Z]] == True

In[23]:= member[INTMUL, map[cart[Z, Z], Z]] := True
```

curry results relating INTMUL and INTTIMES

Theorem.

```
In[24]:= ApComp[composite[IMAGE[inverse[ASSOC]], IMAGE[cross[Id, inverse[E]]]],
           id[range[CURRY]], INTTIMES] // Reverse

Out[24]= APPLY[inverse[CURRY], INTTIMES] == INTMUL

In[25]:= APPLY[inverse[CURRY], INTTIMES] := INTMUL
```

Theorem.

```
In[26]:= ApComp[CURRY, inverse[CURRY], INTTIMES]

Out[26]= APPLY[CURRY, INTMUL] == INTTIMES

In[27]:= APPLY[CURRY, INTMUL] := INTTIMES
```

eval formula

```
In[28]:= Assoc[composite[inverse[SINGLETON], IMG], cross[INTTIMES, SINGLETON], RIGHT[x]]

Out[28]= composite[eval[x], INTTIMES] == composite[INTMUL, RIGHT[x]]

In[29]:= composite[eval[x_], INTTIMES] := composite[INTMUL, RIGHT[x]]
```

the integer divisibility relation

The integer divisibility relation **INTDIV** was early on defined directly in terms of binary homomorphisms.

```
In[30]:= U[binhom[INTADD, INTADD]]

Out[30]= INTDIV
```

Traditionally, divisibility is defined in terms of multiplication. This connection between multiplication and divisibility is now a theorem, which can be derived quickly as follows:

```
In[31]:= Assoc[rotate[E], cross[INTTIMES, Id], inverse[SECOND]] // Reverse

Out[31]= composite[INTMUL, inverse[SECOND]] == INTDIV

In[32]:= composite[INTMUL, inverse[SECOND]] := INTDIV
```

Comment. The properties of the integer divisibility relation **INTDIV** were derived before the introduction of either of the functions **INTTIMES** or **INTMUL**. This non-traditional order of introducing the main concepts of integer arithmetic has been done to facilitate the derivations of the theorems.