MIXMUL, part 1. Basics.

Johan G. F. Belinfante
2006 December 6

```
In[1]:= SetDirectory["l:"]; << goedel88.05a; << tools.m

        :Package Title: goedel88.05a          2006 December 5 at 3:45 p.m.

        It is now:  2006 Dec 7 at 12:11

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 December 5

        weightlimit = 40
```

summary

In this notebook, the binary function **MIXMUL** for mixed multiplication of integers by natural numbers yielding integers is introduced, and some of its basic properties are derived. Traditionally multiplication is often defined in terms of iterated addition, but if one does so, then one must deal with the fairly complex details of the iteration process every time that the definition of multiplication is accessed, dramatically slowing down all applications of the Gödel algorithm for expressions involving multiplication. For this reason, mixed multiplication is here defined by its properties, rather than by a specific construction. Specifically, an integer **z** is defined to be the product of an integer **x** and a natural number **y** if there exists an addition-preserving mapping **w** from the set **omega** of natural numbers to the set **Z** of integers which takes the natural number **1 = set[0]** to the integer **x** and takes the natural number **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[MIXMUL]]]]

Out[3]= class[u_, member[v_, MIXMUL]] := ReleaseHold[Module[{w = Unique[], x =
          Unique[], y = Unique[], z = Unique[]}, class[u, exists[w, x, y, z, and[
          member[w, binhom[NATADD, INTADD]]], equal[v, pair[pair[x, y], z]], equal[
          set[x], image[w, set[set[0]]]], equal[set[z], image[w, set[y]]]]]]]]]
```

normalization for MIXMUL

The function **MIXMUL** can be related to **MIXTIMES** as follows:

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

Out[4]= composite[inverse[SINGLETON], IMG, cross[MIXTIMES, SINGLETON]] == MIXMUL
```

```
In[5]:= composite[inverse[SINGLETON], IMG, cross[MIXTIMES, SINGLETON]] := MIXMUL
```

The definition of mixed multiplication avoids explicit mention of iteration, but iteration does enter directly or indirectly into certain proofs. The idea is to speed up applications of Gödel's algorithm by defining multiplication by its properties, relegating the role of iteration to the proofs of some existence and uniqueness theorems. In particular, the uniqueness theorem for iteration was used to show that **MIXTIMES** is a function, and from that fact it now follows immediately that **MIXMUL** is also a function.

```
In[6]:= SubstTest[FUNCTION, composite[funpart[x], funpart[y]],
          {x -> composite[inverse[SINGLETON], IMG], y -> cross[MIXTIMES, SINGLETON]}] // Reverse

Out[6]= FUNCTION[MIXMUL] == True

In[7]:= FUNCTION[MIXMUL] := True
```

domains of binary homomorphisms

Lemma. (The function **IMAGE[FIRST]** assigns to each set its domain. Since all members of **map[x,y]** have the same domain **x,** the restriction of **IMAGE[FIRST]** to **map[x,y]** is a constant function.)

```
In[8]:= equal[composite[IMAGE[FIRST], id[map[x, y]]], cart[map[x, y], set[x]]] // AssertTest

Out[8]= equal[cart[map[x, y], set[x]], composite[IMAGE[FIRST], id[map[x, y]]]] == True

In[9]:= composite[IMAGE[FIRST], id[map[x_, y_]]] := cart[map[x, y], set[x]]
```

Lemma. (An equation recognized to be true is made into a new rewrite rule.)

```
In[10]:= equal[intersection[binhom[x, y], map[fix[domain[x]], fix[domain[y]]]], binhom[x, y]]

Out[10]= True

In[11]:= intersection[binhom[x_, y_], map[fix[domain[x_]], fix[domain[y_]]]] := binhom[x, y]
```

Theorem. (All binary homomorphisms from **x** to **y** have the same domain **fix[domain[x]]**.)

```
In[12]:= Assoc[IMAGE[FIRST], id[map[fix[domain[x]], fix[domain[y]]]], id[binhom[x, y]]]

Out[12]= composite[IMAGE[FIRST], id[binhom[x, y]]] == cart[binhom[x, y], set[fix[domain[x]]]]

In[13]:= composite[IMAGE[FIRST], id[binhom[x_, y_]]] := cart[binhom[x, y], set[fix[domain[x]]]]
```

FUNPART theorem

Theorem.

```
In[14]:= composite[FUNPART, id[binhom[x, y]]] // ReifNormality

Out[14]= composite[FUNPART, id[binhom[x, y]]] == id[binhom[x, y]]

In[15]:= composite[FUNPART, id[binhom[x_, y_]]] := id[binhom[x, y]]
```

Corollary.

```
In[16]:= Assoc[FUNPART, id[range[MIXTIMES]], MIXTIMES]

Out[16]= composite[FUNPART, MIXTIMES] == MIXTIMES

In[17]:= composite[FUNPART, MIXTIMES] := MIXTIMES
```

domain of MIXTIMES

Theorem.

```
In[18]:= Assoc[composite[IMAGE[FIRST]], id[binhom[NATADD, INTADD]], inverse[eval[set[0]]]]

Out[18]= composite[IMAGE[FIRST], MIXTIMES] == cart[Z, set[omega]]

In[19]:= composite[IMAGE[FIRST], MIXTIMES] := cart[Z, set[omega]]
```

Corollary.

```
In[20]:= IminComp[composite[inverse[SINGLETON], IMG], cross[MIXTIMES, SINGLETON], V]

Out[20]= domain[MIXMUL] == cart[Z, omega]

In[21]:= domain[MIXMUL] := cart[Z, omega]
```

Corollary.

```
In[22]:= Assoc[MIXMUL, id[domain[MIXMUL]], id[cart[V, V]]] // Reverse

Out[22]= composite[MIXMUL, id[cart[V, V]]] == MIXMUL

In[23]:= composite[MIXMUL, id[cart[V, V]]] := MIXMUL
```

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

```
In[24]:= member[MIXMUL, V] // AssertTest

Out[24]= member[MIXMUL, V] == True

In[25]:= member[MIXMUL, V] := True
```

curry results relating MIXMUL and MIXTIMES

Theorem. (A variant of the formula connecting **MIXMUL** with **MIXTIMES**.)

```
In[26]:= Assoc[rotate[E], cross[FUNPART, Id], cross[MIXTIMES, Id]]

Out[26]= composite[rotate[E], cross[MIXTIMES, Id]] == MIXMUL

In[27]:= composite[rotate[E], cross[MIXTIMES, Id]] := MIXMUL
```

Theorem. (Yet another variant.)

```
In[28]:= (composite[rotate[E], cross[x, Id]] // TripleRotate // Reverse) /. x → MIXTIMES

Out[28]= rotate[composite[inverse[MIXTIMES], E]] == MIXMUL

In[29]:= rotate[composite[inverse[MIXTIMES], E]] := MIXMUL
```

Theorem.

```
In[30]:= ApComp[composite[IMAGE[inverse[ASSOC]], IMAGE[cross[Id, inverse[E]]]],
           id[range[CURRY]], MIXTIMES] // Reverse

Out[30]= APPLY[inverse[CURRY], MIXTIMES] == MIXMUL

In[31]:= APPLY[inverse[CURRY], MIXTIMES] := MIXMUL
```

Theorem.

```
In[32]:= ApComp[CURRY, inverse[CURRY], MIXTIMES]

Out[32]= APPLY[CURRY, MIXMUL] == MIXTIMES

In[33]:= APPLY[CURRY, MIXMUL] := MIXTIMES
```

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

```
In[34]:= SubstTest[implies, member[w, map[x, map[y, z]]],
           or[empty[y], member[APPLY[inverse[CURRY], w], map[cart[x, y], z]]],
           {w → MIXTIMES, x → Z, y → omega, z → Z}] // Reverse

Out[34]= member[MIXMUL, map[cart[Z, omega], Z]] == True

In[35]:= member[MIXMUL, map[cart[Z, omega], Z]] := True
```

Corollary. (Composite with **eval[x]**.)

```
In[36]:= SubstTest[composite, eval[x],
           APPLY[CURRY, composite[funpart[setpart[y]], id[cart[V, V]]]], y → MIXMUL] // Reverse

Out[36]= composite[eval[x], MIXTIMES] == composite[MIXMUL, RIGHT[x]]
```

```
In[37]:= composite[eval[x_], MIXTIMES] := composite[MIXMUL, RIGHT[x]]
```

range

Lemma. (Corollary of the mapping property derived in the preceding section.)

```
In[38]:= SubstTest[implies, member[w, map[x, y]],
           subclass[range[w], y], {w → MIXMUL, x → cart[Z, omega], y → Z}] // Reverse

Out[38]= subclass[range[MIXMUL], Z] == True

In[39]:= % /. Equal → SetDelayed
```

Lemma. (A consequence of the fact that **eval[x]** is a function.)

```
In[40]:= Map[inverse, SubstTest[composite, funpart[x], id[y],
            inverse[funpart[x]], {x → eval[set[0]], y → binhom[NATADD, INTADD]}]] // Reverse

Out[40]= composite[MIXMUL, RIGHT[set[0]]] == id[Z]

In[41]:= composite[MIXMUL, RIGHT[set[0]]] := id[Z]
```

Lemma. (Any image provides a lower bound for the range.)

```
In[42]:= Map[subclass[#, range[MIXMUL]] &, ImageComp[MIXMUL, RIGHT[set[0]], V]]

Out[42]= subclass[Z, range[MIXMUL]] == True

In[43]:= % /. Equal → SetDelayed
```

Theorem. (Equation for the range of **MIXMUL**.)

```
In[44]:= SubstTest[and, subclass[x, y], subclass[y, x], {x → range[MIXMUL], y → Z}]

Out[44]= equal[Z, range[MIXMUL]] == True

In[45]:= range[MIXMUL] := Z
```

connection with MIXDIV

The mixed divisibility relation **MIXDIV** was defined directly in terms of binary homomorphisms before mixed multiplication had been introduced:

```
In[46]:= U[binhom[NATADD, INTADD]]

Out[46]= MIXDIV
```

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

```
In[47]:= Assoc[rotate[E], cross[MIXTIMES, Id], inverse[SECOND]] // Reverse

Out[47]= composite[MIXMUL, inverse[SECOND]] == MIXDIV

In[48]:= composite[MIXMUL, inverse[SECOND]] := MIXDIV
```